uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2017-2018

Trabajo Fin de Grado

"Predicción de Radiación Solar a partir de Modelos Físicos y Aprendizaje Automático"

Ismael Francisco Mendaña

Tutor

Javier Huertas Tato

Leganés, 2018

Agradecimientos

Quisiera agradecer a mi familia por el apoyo que han supuesto durante este largo viaje, por darme ánimos durante todo este tiempo.

También quiero agradecer a mis amigos, por hacer que los momentos para relajarse y desconectar sean únicos, y darme fuerzas para continuar.

Por último, pero no menos importante, quisiera hacer dos menciones especiales. A mi compañero de prácticas y amigo, Fernando, por aguantarme en tantas prácticas y aun así seguir apoyándome. Y a mi tutor, Javier, por soportarme durante este viaje que ha sido la realización de este trabajo.

Gracias.

Resumen

Este proyecto es un estudio para comprobar una nueva aproximación a un problema en particular, la predicción de la radiación solar global. La radiación solar es principalmente usada para calcular la producción solar de las plantas fotovoltaicas, por ello es importante ser capaz de determinar en un momento dado su valor, poder hacer estimaciones de producción, y poder mantener el equilibrio en la red eléctrica entre consumo y producción.

En este estudio se intenta probar si la aplicación de algoritmos de inteligencia artificial mejora los resultados obtenidos por los modelos físicos tradicionales empleados en la predicción de la radiación solar.

Para llevar a cabo este estudio se ha utilizado un conjunto de datos al cual se le han aplicado las transformaciones necesarias para crear un modelo base, y el modelo a probar. Posteriormente se han aplicado técnicas de aprendizaje automático, usando diferentes algoritmos para ver la eficacia a la hora de predecir los resultados, y ver si existe una mejora sobre el caso base del modelo que se presenta.

Por último, se han analizado los resultados para ver si realmente existe una mejora. Estos han sido bastante positivos ya que se han podido observar mejoras con respecto al modelo general de referencia. Por lo que se pueden considerar satisfactorios. Una explicación más en detalle será desarrollada en el documento.

Palabras clave: Inteligencia artificial, aprendizaje automático, radiación solar, problema de regresión.

Abstract

This project is a study to test a new approach to the Global Horizontal Irradiance forecasting problem. Solar radiation is mainly used to calculate the production of the photovoltaic power stations, that is why it is important to be able to determine its value at a given horizon, to be able to calculate estimates of production, and maintain the balance in the electricity network between consumption and production.

In this study we will test whether the application of artificial intelligence algorithms improves the results obtained by the traditional physical models used in the prediction of solar irradiation.

To perform this study, a dataset has been used which has been applied the necessary transformations to create a base model, and the model to be tested. After that, machine learning techniques have been applied, using different algorithms to check the effectiveness of the predicted results, and see if there is an improvement over the base model with the presented model.

Finally, the results have been analysed to see if there really is an improvement. These have been quite positive since improvements have been observed with respect to the general reference model. For what can be considered satisfactory. A more detailed explanation will be developed in the document.

Keywords: Artificial intelligence, machine learning, solar irradiance, regression problem.

Índice

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

Capítulo 1

Introducción

En las últimas décadas la tecnología ha experimentado un gran empuje. Cada vez contamos con más potencia de cálculo en los ordenadores y el acceso a los recursos educativos es más fácil gracias a internet. Aunque el campo de la inteligencia artificial no es nuevo, es en estos últimos años cuando está experimentando un gran crecimiento, por lo que es un buen momento para buscar aproximaciones diferentes a problemas actuales.

En este caso el problema es la predicción de la radiación solar. Esto está relacionado con la capacidad de producción de energía de las plantas de paneles fotovoltaicos, ya que, si se es capaz de predecir la radiación, se podrían de hacer estimaciones de producción. Actualmente esto se hace mediante el uso de modelos físicos, por lo que se va a intentar aplicar las técnicas de aprendizaje automático, como la validación cruzada o el ajuste de hiperparámetros entre otras, sobre estos modelos para ver si se puede obtener mejores resultados en la predicción.

Ilustración 1: Paneles Solares

En este documento se van a detallar los pasos seguidos durante el proceso para realizar esta investigación, así como las técnicas aplicadas, consideraciones hechas, resultados obtenidos, etc. Este estudio de carácter práctico por lo que se ha desarrollado una aplicación para la realización de todos los cálculos necesarios.

A continuación, se presenta la estructura del documento en los siguientes capítulos del documento:

- **Capítulo 2 – Estado de la cuestión:** En este capítulo se procederá a explicar la problemática actual, así como una descripción de en qué consisten los modelos físicos. También se verán otras aproximaciones diferentes a la planteada y explicaremos cuales han sido los algoritmos de aprendizaje automático que han sido objeto de estudio.
- **Capítulo 3 – Objetivos:** En este apartado se expondrá la lista de objetivos que se buscan lograr con este desarrollo y también su posible impacto socioeconómico.
- **Capítulo 4 – Desarrollo:** En este capítulo se explicará el grueso del detalle sobre cómo se ha llevado a cabo el proyecto, haciendo explicaciones detalladas sobre

cómo se ha realizado, las decisiones tomadas y todos los pasos seguidos para alcanzar el resultado.

- **Capítulo 5 – Resultados:** En este capítulo se agruparán los resultados obtenidos de la investigación y se mostrarán gráficamente.
- **Capítulo 6 – Conclusiones y líneas futuras:** En este apartado se expondrán las conclusiones obtenidas del análisis de los resultados, y las líneas futuras que podrían seguirse, así como posibles mejoras en la investigación.

Capítulo 2

Estado de la cuestión

En los últimos años cada vez la gente dispone de más aparatos electrónicos, desde móviles, a electrodomésticos, o incluso coches. Cada año se ve incrementada la demanda energética, como se puede ver en la siguiente ilustración que refleja la demanda eléctrica en España [1].

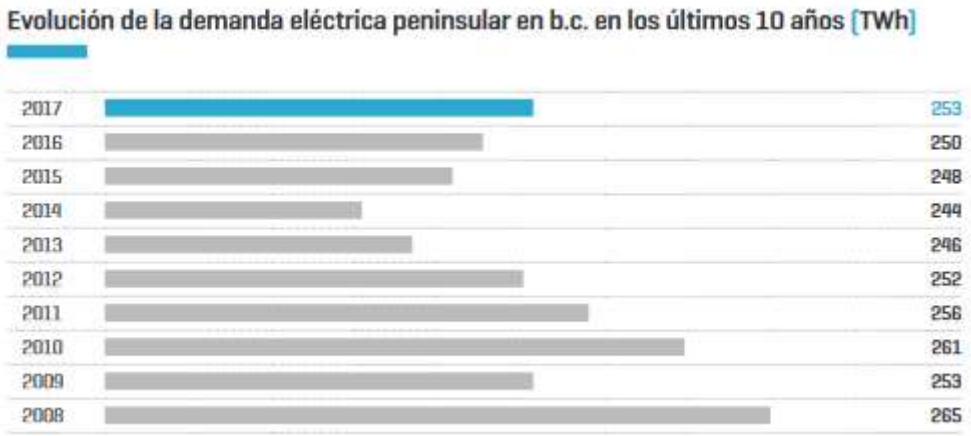

Ilustración 2: Demanda eléctrica de 2017 en España según Red Electrica Española [1]

Sin tener en cuenta el desplome producido durante los años de la crisis, se puede ver que la demanda energética va en aumento. Esto supone nuevos retos ya que cada vez se tiene más concienciación sobre el medio ambiente, por lo que debería intentar que la mayor cantidad de producción de energía corresponda a medios sostenibles. Esto implica apostar por las energías renovables.

En España las energías renovables representan un 46% de la potencia instalada en el conjunto del parque generador [2], y en 2017 supuso un 33,7% de la producción total de energía durante el año [2]. Esto enseña que aún se tiene un largo camino que recorrer hasta que se tenga una mejor cuota en la producción eléctrica de forma sostenible.

Energía solar

Para este estudio interesa, principalmente, una sola fuente de energía renovable, la energía solar fotovoltaica. La energía solar todavía representa un pequeño porcentaje de la producción de energías renovables, ya que solo produce algo más del 5% del total de energía generada, según datos de 2017 [2]. España es un país que cuenta con muchas horas de sol a lo largo del año, por eso este tipo de energía se podría aprovechar en mayor medida si se incrementase el parque instalado.

En la siguiente ilustración se puede ver cómo está repartido el parque solar en España, según datos de 2017 [2]:

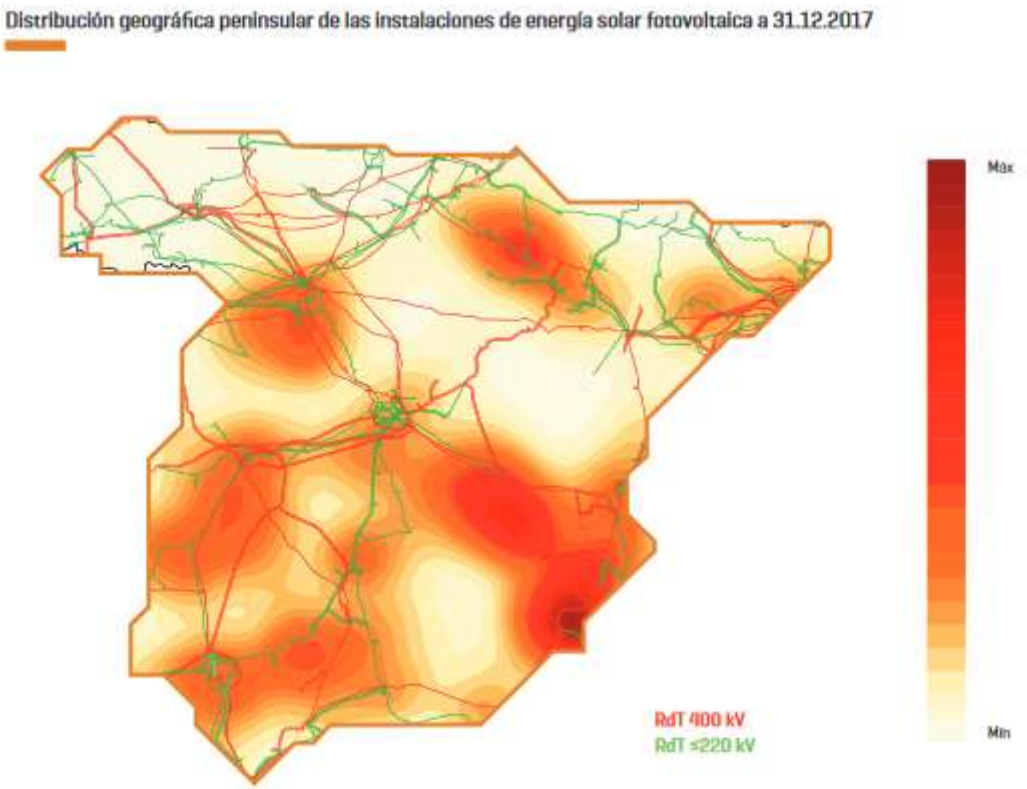

Ilustración 3: Distribución del parque solar instalado en 2017 [2]

Para entender mejor donde encaja el estudio que se presenta en este documento, primero se tiene que entender donde encaja la radiación solar en la cadena de producción y consumo eléctrico.

Radiación solar y energía fotovoltaica

La radiación solar es el conjunto de ondas electromagnéticas emitidas por el sol. Esta radiación puede ser medida en el espacio o en la superficie terrestre. Si es medida en la superficie terrestre hay factores que afectan a su valor, como en qué posición se encuentra el sol en ese momento, o las condiciones climáticas. Existen varios tipos de mediciones sobre la radiación solar, pero este estudio se va a centrar en la radiación solar global, (GHI por sus siglas en inglés de Global Horizontal Irradiance).

Ahora tenemos que ver qué importancia tiene este concepto en la producción de energía eléctrica. Ahí es donde entra la energía solar fotovoltaica. Esta energía proviene de la transformación de la radiación solar en una corriente eléctrica usando la tecnología fotovoltaica [3].

Teniendo claros estos conceptos, se puede ver la relevancia que tiene este campo de estudio. Si somos capaces de predecir la radiación solar, se puede determinar la cantidad de energía que se produce en una planta solar fotovoltaica.

Aproximaciones anteriores

Como se ha comentado anteriormente ya hay aproximaciones similares a la que vamos a llevar acabo en este proyecto, como se puede ver en [4]. El estudio que se va a llevar a cabo en este documento tiene como base el texto citado anteriormente. Se usan datos similares, pero el modelo a probar es diferente a los propuestos en ese estudio.

Otra de las aproximaciones más utilizadas es el uso de modelos físicos para realizar las predicciones de la radiación solar. Estos modelos, por lo general, suelen ser conjuntos de ecuaciones diferenciales que son resueltas mediante métodos numéricos. Los datos que se han utilizado para hacer las pruebas han sido recogidos en una estación de mediciones de Sevilla. Los datos representan 2 años de mediciones y observaciones recogidas durante los años 2015 a 2107.

Los datos recogidos, explicados en su punto correspondiente de la memoria, son los datos producidos por cuatro modelos físicos. A continuación, procedemos a explicar en qué consiste cada uno de estos modelos.

Modelos físicos

Los datos usados como entradas en este estudio son las predicciones realizadas por cuatro modelos físicos, o también llamados predictores, estos son los 4 utilizados:

- ***Satellite Method*:** Este modelo consiste en la utilización de imágenes por satélite para predecir la radiación solar global [5]. Se comparan dos imágenes consecutivas capturadas por satélite, y se calcula un vector de trayectoria de las nubes que pueda haber en la imagen. Con esta información se determina si el punto de interés a medir está en la trayectoria de esas nubes. Usando esta estimación, y datos de mediciones en tierra, se determina la radiación solar global.

- ***WRF-Solar*:** Este modelo es una ampliación del modelo llamado *Weather Research and Forecasting.* Se basa en modelos matemáticos para predecir la climatología según las condiciones de la atmosfera y los océanos, que combinada junto con modelos de interacción de nubes y partículas es capaz predecir la radiación [6].

- ***CIADCast*:** Este modelo llamado *Cloud Index Advection and Diffusion* [7], transforma los datos obtenidos por el satélite *Meteosat*, usando el modelo *WRF* para obtener el coeficiente de cobertura efectiva de nubes, que es transformado posteriormente en una medición de radiación solar usando el método *Heliosat-2* [8].

- ***Smart Persistence*:** Este es el modelo más simple de los cuatro presentados. Se toman los valores de la radiación solar actual y la radicación solar con cielo despejado en el instante inicial de la predicción que se quiere realizar. Se realiza una división entre estos valores para obtener el coeficiente de cielo despejado, y se multiplica por la radiación solar con cielo despejado en el momento que se quiere predecir [9]. La fórmula sería de la siguiente manera (Eq. 1):

$$f(t) = \frac{RS_{actual}}{RS_{CieloDespejado}(0)} * RS_{CieloDespejado}(t) \quad (1)$$

Ecuación 1: Fórmula modelo de Persistencia Inteligente

Aproximación del Estudio

Ahora se va a presentar la propuesta para el nuevo modelo que se va a desarrollar a lo largo de este documento. Como ya se indicó anteriormente esta aproximación es totalmente diferente a las expuestas en [4]. Lo que si comparten ambos estudios es el conjunto de datos de prueba, y la aproximación general, para poder tener un punto de partida sobre el que trabajar.

Modelo general

El modelo general consiste en usar como entradas para realizar la predicción los valores que los modelos físicos han predicho para ese horizonte. Esto queda representado en la siguiente ecuación (Eq. 2):

$$f(t,h) = f(P_1(t,h), P_2(t,h), P_3(t,h), P_4(t,h)) \quad (2)$$

Ecuación 2: Modelo General

Donde t es el instante de tiempo para la predicción, h es el horizonte sobre el que se quiere hacer la predicción, P_x siendo x entre 1 y 4, cada uno correspondiente con uno de los modelos físicos. Con un ejemplo para explicarlo de una forma más sencilla, si se quiere, en el instante t=0, hacer una predicción en el horizonte h=15 la fórmula sería la siguiente (Eq. 3):

$$f(0,15) = f(P_1(0,15), P_2(0,15), P_3(0,15), P_4(0,15)) \quad (3)$$

Ecuación 3: Ejemplo del modelo general con el horizonte 15

Modelo por vecindad

El modelo por vecindad consiste en usar los valores de los horizontes anterior y posterior de cada uno de los modelos físicos para complementar los datos en ese predictor, es decir la fórmula general de la predicción sería (Eq. 4):

$$f(t,h) = f\begin{pmatrix} P_1(t,h-15), P_1(t,h), P_1(t,h+15), \dots, \\ P_4(t,h-15), P_4(t,h), P_4(t,h+15) \end{pmatrix} \quad (4)$$

Ecuación 4: Modelo por vecindad

Donde t es el instante de tiempo para la predicción, h es el horizonte sobre el que se quiere hacer la predicción, P_x siendo x entre 1 y 4, cada uno correspondiente con uno de los modelos físicos. Con un ejemplo para explicarlo de una forma más sencilla si se quiere, en el instante t=0, hacer una predicción en el horizonte h=15 la fórmula sería la siguiente (Eq. 5):

$$f(0{,}15) = f\big(P_1(0,0), P_1(0,15), P_1(0{,}30), \dots, P_4(0{,}0), P_4(0{,}15), P_4(0{,}30)\big) \qquad (5)$$

Ecuación 5: Ejemplo del modelo por vecindad con el horizonte igual a 15

Una vez presentados los 2 modelos que se van a comparar, se tiene que explicar cuáles son las técnicas y algoritmos de aprendizaje automático que vamos a emplear para este estudio. Como ya se ha indicado, el estudio comparativo es sobre los modelos, por lo que se delegará lo máximo posible en librerías externas que nos ahorren tiempo de desarrollo y así poder centrarnos más en el estudio en sí.

Aprendizaje Automático

En este punto se desarrollará la explicación sobre las técnicas empleadas, validación cruzada y ajuste de hiperparámetros de los algoritmos. También se explicará que algoritmos se han empleado.

Validación cruzada

Es una técnica ampliamente utilizada para la evaluación de modelos en inteligencia artificial. Se trata de dividir un conjunto de datos en varios subconjuntos e ir iterando y usando esos mismos subconjuntos como datos de entrenamiento o como datos de prueba indistintamente. De cada una de esas iteraciones se extraerán métricas de evaluación, y el resultado final será la media de las n iteraciones realizadas [10].

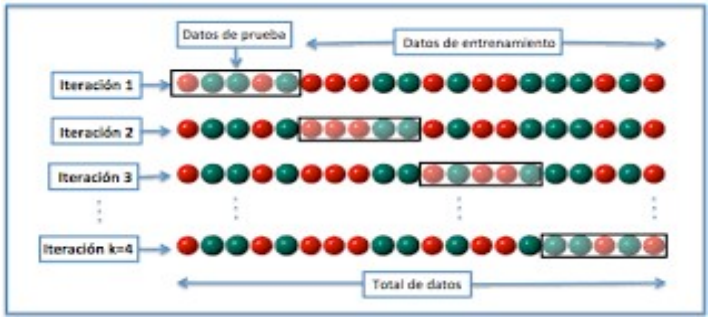

Ilustración 4: Ejemplo validación cruzada con 4 subconjuntos

Como se puede ver en la ilustración 6, el ejemplo muestra una división en 4 iteraciones. Eso quiere decir que el conjunto ha sido dividido en 4, y en cada iteración 3 subconjuntos conforman los datos de entrenamiento, y 1 subconjunto los datos de prueba.

Esta técnica es utilizada principalmente para probar modelos que se quieren llevar a la práctica, ya que permite garantizar la independencia de los resultados respecto a los datos de entrenamiento y prueba [10]. Se puede dar un problema en el aprendizaje supervisado cuando se utiliza el mismo conjunto de datos tanto en el entrenamiento como para las pruebas. Es lo que se conoce como sobreaprendizaje. Esto hace que el algoritmo tenga unos resultados muy buenos, ya que sabe lo que tiene que predecir en los entrenamientos, pero su rendimiento decaiga cuando es expuesto a nuevos datos para predecir.

Existen varios tipos de validación cruzada, en función si se predefinen el número de subconjuntos que se deben formar o no. Para este desarrollo se usará una aproximación propia, ya que así lo requiere el problema. Esta aproximación será explicada en el punto correspondiente más adelante.

Estimación de hiperparámetros

Esta es otra de las técnicas que se han empleado para este desarrollo. Uno de los principales problemas que tienen algunos de los algoritmos de aprendizaje automático es que tienen parámetros que se pueden modificar y afectan a los resultados de las predicciones realizadas. Por ejemplo, uno de los algoritmos usados, el *Random Forest,* tiene como parámetros el número de árboles/nodos que compone el bosque. Estos valores pueden afectar a la precisión de las predicciones, por eso se ha empleado esta técnica. Además, estos hiperparámetros son distintos para cada problema, por lo que cada vez que se presenta un modelo hay que buscar cual es la mejor configuración.

El procedimiento es el siguiente, y se aplica dentro de cada una de las iteraciones de la validación cruzada. En este caso para los algoritmos que lo necesiten en nuestro desarrollo, se ha creado una lista con los valores a introducir en esos parámetros. En lugar de coger el subconjunto que corresponda de entrenamiento y aplicarlo directamente al algoritmo, lo que se hace es, volver a dividir ese subconjunto. En este caso creamos un conjunto de entrenamiento y otro de validación como se refleja en la ilustración 5.

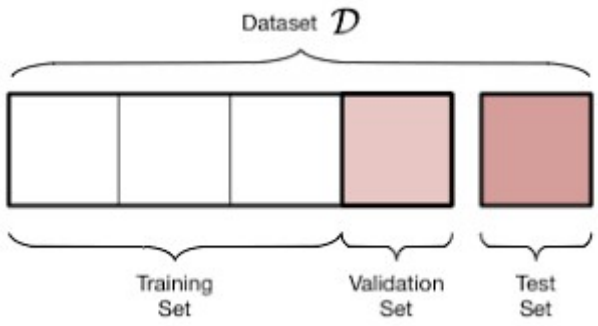

Ilustración 5: División del subconjunto de entrenamiento en entrenamiento y validación

Como ha sido explicado en el punto anterior, tenemos que hacer esta nueva división, porque si usásemos los conjuntos previamente creados, tendríamos un problema de sobreaprendizaje. La división realizada en este caso es aleatoria. Como solo se quieren encontrar los mejores hiperparámetros, en este caso, no merece la pena implementar una separación sofisticada porque el impacto no va a ser tan relevante. Una vez que se tiene la nueva subdivisión se entrena el algoritmo con esos datos, y después se usará el subconjunto de validación para las pruebas. Repetiremos este proceso por cada uno de los valores que se han definido en la lista de parámetros, y finalmente se escogerá el que mejores resultados ha obtenido.

Este proceso se repetirá en cada una de las n iteraciones de la validación cruzada, ya que hay que hacerlo por cada uno de los subconjuntos. Una vez obtenido el mejor hiperparámetro para ese algoritmo, el siguiente paso es, con el conjunto completo de entrenamiento que se había definido para esa iteración de la validación cruzada, se entrenará el algoritmo, y con el subconjunto de pruebas se realizaran las predicciones, aplicando el parámetro obtenido en este proceso al algoritmo en cuestión. Esto asegura que se evita la

sobreaprendizaje, y que se está entrenando el algoritmo con el mejor conjunto de hiperparámetros.

Estas son las 2 técnicas que se han empleado principalmente. En el siguiente apartado se explicará la naturaleza del problema y los algoritmos empleados para realizar las pruebas.

Algoritmos

Llegados a este punto, después de haber puesto todo en contexto, se va a explicar que algoritmos se van a utilizar para las pruebas. Pero antes se debe definir el problema que se tiene, para poder escoger con mejor criterio dichos algoritmos.

Para poder determinar de qué tipo es el problema, se deben analizar los datos que se tienen, y el objetivo. Observando los datos, se puede ver que se tiene una serie temporal, sobre la que queremos realizar predicciones. Es decir, se quiere establecer la relación de dependencia que existe con nuestras variables para poder realizar predicciones. Esto se corresponde con un análisis de regresión.

Esto es importante porque no todos los algoritmos de aprendizaje automático resuelven el mismo tipo de problemas. Por ejemplo, hay algoritmos que resuelven mejor los problemas de clasificación.

Para resolver este problema de regresión, se han seleccionado 4 algoritmos, que se explicarán a continuación:

- **Regresión lineal:** Es un modelo matemático que permite aproximar la relación de dependencia entre 2 o más variables. La ecuación que expresa esta relación sería la mostrada a continuación (Eq. 6):

$$f(t) = C_0 + C_1P_1 + \dots + C_nP_n \quad (6)$$

Ecuación 6: Regresión lineal

  Donde C_n representan los coeficientes a calcular que determinan la dependencia de las variables. P_n representa las variables, en nuestro caso los predictores. C_o es un coeficiente especial llamado intersección.

- **Máquina de vectores de soporte:** (por sus siglas en inglés SVM, *support vector machine*). Este algoritmo representa los valores en un espacio dimensional, e intenta buscar el hiperplano, o hiperplanos que mejor separan los datos [11]. La construcción de esos hiperplanos viene determinada por una función principal, o también conocida como *kernel*. Aunque esta definición se ajusta más a una clasificación, también puede ser utilizado en problemas de regresión, como se indica en [12]. Para este problema se han probado 2 *kernels* diferentes, uno lineal y otro de base radial Gaussiana. Se puede ver un ejemplo en la ilustración 6.

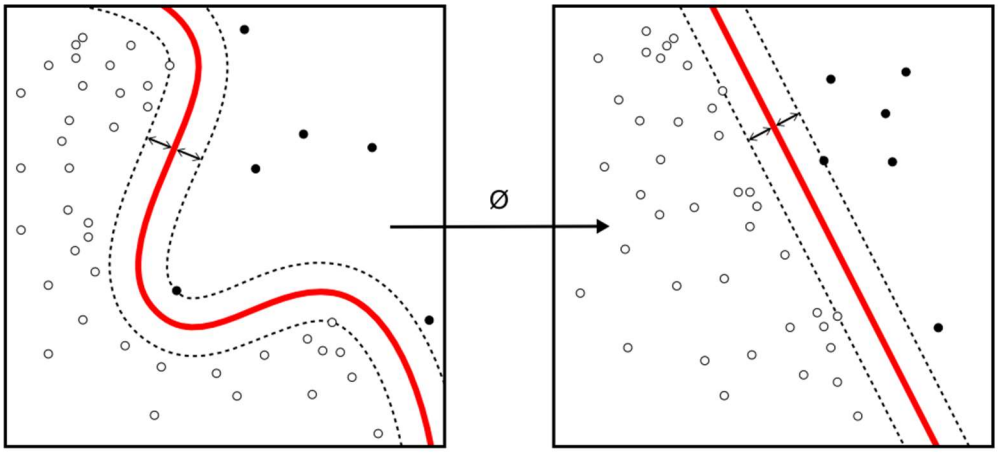

Ilustración 6: Ejemplo SVM

- **Random Forest:** Este algoritmo construye múltiples arboles de decisión que entrena de forma independiente para realizar la predicción. Un árbol de decisión es una estructura en la que cada nodo representa una decisión a tomar. Cuantos más nodos más posibles resultados finales se pueden obtener. En la ilustración 7 se puede ver un ejemplo.

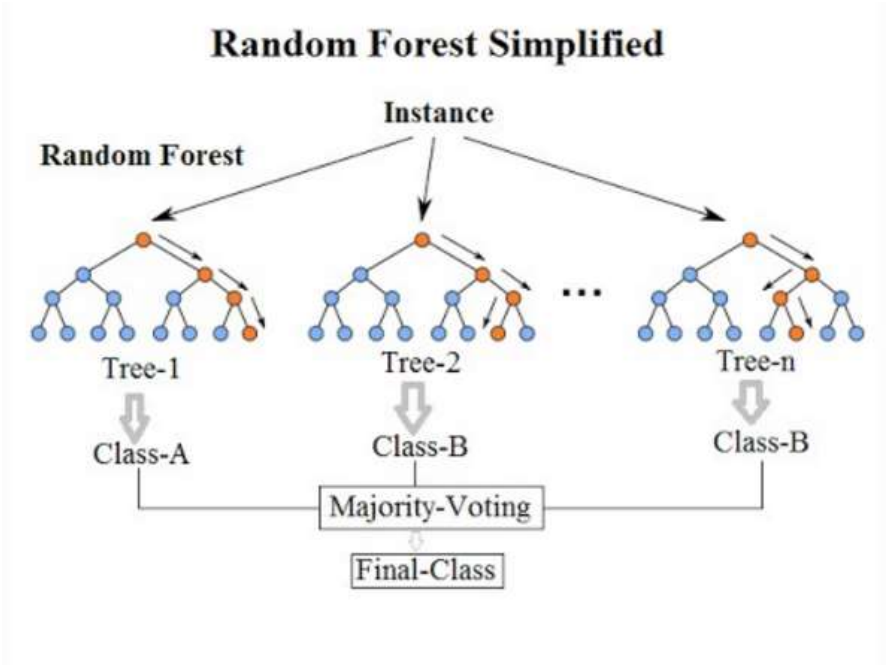

Ilustración 7: Ejemplo Random Forest

Cada uno de los árboles que componen el bosque se entrena con un subconjunto de los datos elegidos al azar. El resultado es el cálculo de la media de cada uno de los resultados de cada árbol de decisión [13].

Estos son los algoritmos utilizados para la obtención de las métricas. Se han usado varios para así poder contrastar los resultados y contar con una mayor cantidad de información para poder valorar el modelo propuesto.

Capítulo 3

Objetivos

Objetivos

El objetivo de este proyecto es ver si el modelo que se presenta es mejor a la hora de predecir la radiación solar que el escenario actual. Para ello se ha usado como caso base un modelo general que consiste en introducir los valores de los modelos físicos a los algoritmos de aprendizaje automático, y compararlo con el resultado del modelo propuesto, un modelo por vecindad. Para ello se deberán cumplir los siguientes objetivos:

- **Lenguaje de programación:** Se deberá usar como lenguaje uno que cumpla las dos siguientes condiciones. Que sea un lenguaje de carácter *open source*, y que tenga un conjunto de librerías, también *open source*, para el manejo de algoritmos de aprendizaje automático.
- **Normalización:** Se deberá normalizar los datos para evitar problemas a la hora de introducirlos en los algoritmos de aprendizaje automático.
- **Validación cruzada:** Se deberá aplicar esta técnica para asegurarnos de que no tenemos un sobreajuste en nuestras predicciones.
- **Ajuste de parámetros:** Se deberá aplicar esta técnica para poder encontrar, en los algoritmos que sea necesario, los parámetros que nos proporcionen los mejores resultados para nuestro supuesto.
- **Entrenamiento de algoritmos:** Una vez realizados los pasos anteriores se entrenarán los algoritmos y se usaran los conjuntos de prueba para ver qué resultado se obtiene en las predicciones.
- **Calculo de errores:** Para comprobar el rendimiento en la predicción se calcularán una serie de errores que utilizaremos como medida para saber que hipótesis ha obtenido mejores resultados.
- **Almacenamiento de resultados:** Se deberán almacenar los resultados obtenidos de manera que puedan ser interpretados y analizados de forma cómoda.

Entorno socio-económico

La principal aplicación que podría tener este estudio sería la mejora de la estimación sobre la producción de las plantas fotovoltaicas. Su posible repercusión a nivel económico sería un mejor ajuste y cálculo de la producción eléctrica, una mejor planificación a medio y corto plazo, que derivaría en una reducción de costes a la hora de generar energía eléctrica, ya que, si somos capaces de predecir la radiación, por ejemplo, se podría compensar una caída en la red de forma más eficiente.

A nivel social, podría repercutir en una disminución del coste al usuario final en la factura de la luz, porque si los procesos son más eficientes y se evita un derroche de recursos, por lo tanto, habría un ahorro que podría repercutir a los usuarios.

Marco regulador

En este apartado se abordará las cuestiones legales sobre este estudio. En primer lugar, se tratará el tema de la licencia del documento entregado. Este documento se presenta licenciada bajo *Creative Commons* (como se indica en la portada). Se ha optado por una versión bastante restrictiva. Solo permite que otros puedan descargar este documento y compartirlo

con otras personas, que siempre que se reconozca mi autoría, no están permitidas las modificaciones de ningún tipo sobre este documento, y no se permite su uso para ningún fin comercial [16].

El otro punto legal es el licenciamiento del lenguaje de programación usado. Python es distribuido bajo licencia *Open Source* compatible con GPL [17]. Esto permitirá desarrollar el *software* sin ningún tipo de regalía o coste.

Por último, la licencia de la librería utilizada como apoyo en este proyecto, *Scikit-Learn,* también es *Open Source* y está distribuida bajo la licencia BSD [15], así que al igual que en el punto anterior, nos está permitido su uso sin ningún tipo de coste adicional.

Capítulo 4

Desarrollo

En este capítulo se describirán todas las fases seguidas hasta la obtención de resultados finales de la aplicación creada. Se explicarán la naturaleza de los datos de entrada, así como los pasos que realiza el sistema para llegar a la solución final. Para ello este apartado se va a dividir en los siguientes apartados:

- **Análisis del sistema:** En este apartado extraeremos los requisitos del sistema en base al texto descrito en el apartado de los objetivos, y con algunas restricciones adicionales.
- **Toma de decisiones:** En este apartado se explicarán ciertas decisiones tomadas de carácter técnico para la implementación del problema.
- **Descripción de los datos:** En este apartado se explicará en que campos consiste el conjunto de datos de entrada, y qué importancia tiene cada uno de los campos.
- **Descripción del desarrollo:** En este apartado se irá describiendo paso a paso las acciones que realiza el sistema para llegar a los resultados, explicando cada una de las técnicas usadas y su adaptación a esta aplicación.

Análisis del sistema

Antes de resolver el problema comenzaremos por dividirlo en partes más pequeñas para así simplificar su implementación. Para ello comenzaremos con los requisitos del sistema, que se han extraído de los objetivos del proyecto.

Los requisitos los podremos encontrar divididos en 2 tipos diferentes:

- **Requisitos generales:** Requisitos que afecta de forma general a la aplicación.
- **Requisitos de implementación:** Requisitos por los que se ve afectada la aplicación.

Para recoger los requisitos de forma más ordenada, se mostrarán en una tabla como la del ejemplo:

Código-No/Funcional-Número	
Nombre	Nombre del requisito
Descripción	Descripción del requisito
Prioridad	Prioridad del requisito: Alta, Media, Baja

Tabla 1: Ejemplo de requisito

Cada una de las tablas estará compuesta por 4 campos que contendrán la siguiente información:

- **Identificador de requisito:** Este identificador único estará compuesto de un código (GEN o IMPL) que identificará si el requisito es general o de implementación. Después si el requisito es funcional o no funcional (FUN o NOFUN), y por último un código numérico para identificarlo.
- **Nombre:** Un nombre corto para llamar al requisito.
- **Descripción:** Una breve descripción de en qué consiste el requisito.
- **Prioridad:** Una prioridad en su cumplimiento (Alta, Media o Baja).

Requisitos generales

GEN-FUN-01	
Nombre	Lectura de Datos
Descripción	El sistema debe ser capaz de leer los datos del fichero de origen en formato CSV y almacenarlos en memoria para poder tratarlos
Prioridad	Alta

Tabla 2: Requisito GEN-FUN-01

GEN-FUN-02	
Nombre	Transformación de los datos
Descripción	El sistema debe ser capaz de transformar los datos de entrada conforme al modelo de vecindad
Prioridad	Alta

Tabla 3: Requisito GEN-FUN-02

GEN-FUN-03	
Nombre	Normalización de los datos
Descripción	El sistema debe ser capaz de normalizar los datos conforme a los parámetros que se le proporcionen
Prioridad	Alta

Tabla 4: Requisito GEN-FUN-03

GEN-FUN-04	
Nombre	Desnormalización de los datos
Descripción	El sistema debe ser capaz de deshacer el proceso de normalización para poder comparar los datos
Prioridad	Alta

Tabla 5: Requisito GEN-FUN-04

GEN-FUN-05	
Nombre	Validación Cruzada
Descripción	El sistema debe ser capaz de aplicar la técnica de validación cruzada a los datos según el criterio que se especifique.
Prioridad	Alta

Tabla 6: Requisito GEN-FUN-05

GEN-FUN-06	
Nombre	Ajuste de parámetros
Descripción	El sistema debe ser capaz de realizar un ajuste en los parámetros de entrenamiento de los algoritmos que así lo requieran.
Prioridad	Alta

Tabla 7: Requisito GEN-FUN-06

GEN-FUN-07	
Nombre	Entrenamiento de algoritmos
Descripción	El sistema debe ser capaz de entrenar diferentes algoritmos para realizar las pruebas.
Prioridad	Alta

Tabla 8: Requisito GEN-FUN-07

GEN-FUN-08	
Nombre	Cálculo del error nMAE
Descripción	El sistema debe ser capaz de realizar los cálculos necesarios para obtener el error de predicción nMAE
Prioridad	Alta

Tabla 9: Requisito GEN-FUN-08

GEN-FUN-09	
Nombre	Cálculo del error nRMSE
Descripción	El sistema debe ser capaz de realizar los cálculos necesarios para obtener el error de predicción nRMSE
Prioridad	Alta

Tabla 10: Requisito GEN-FUN-09

GEN-FUN-10	
Nombre	Cálculo del error MAE
Descripción	El sistema debe ser capaz de realizar los cálculos necesarios para obtener el error de predicción MAE
Prioridad	Alta

Tabla 11: Requisito GEN-FUN-10

GEN-FUN-11	
Nombre	Cálculo del error RMSE
Descripción	El sistema debe ser capaz de realizar los cálculos necesarios para obtener el error de predicción RMSE
Prioridad	Alta

Tabla 12: Requisito GEN-FUN-11

GEN-FUN-12	
Nombre	Almacenamiento de Resultados
Descripción	El sistema debe ser capaz de ser capaz de almacenar los resultados en un fichero de texto en formato CSV
Prioridad	Alta

Tabla 13: Requisito GEN-FUN-12

Requisitos de implementación

IMPL-FUN-01	
Nombre	Ejecución aplicación
Descripción	El sistema debe ser capaz de ejecutarse en segundo plano, como una aplicación de consola.
Prioridad	Media

Tabla 14: Requisito IMPL-FUN-01

IMPL-NOFUN-01	
Nombre	Sistema Operativo
Descripción	El sistema debe ser capaz de funcionar en entornos Windows o Linux independientemente.
Prioridad	Baja

Tabla 15: Requisito IMPL-NOFUN-01

IMPL-NOFUN-02	
Nombre	Lenguaje de Programación
Descripción	El sistema debe ser desarrollado en un lenguaje *Open Source*, que posea un conjunto de librerías potente para desarrollo de aplicaciones de aprendizaje automático.
Prioridad	Alta

Tabla 16: Requisito IMPL-NOFUN-02

IMPL-NOFUN-03	
Nombre	Librerías de Aprendizaje Automático
Descripción	El sistema debe ser capaz hacer uso de alguna librería *Open Source* que tenga ya implementados los algoritmos para ser utilizados.
Prioridad	Alta

Tabla 17: Requisito IMPL-NOFUN-03

Toma de decisiones

En este apartado se explicarán algunas de las decisiones que se han tomado, principalmente sobre que lenguaje de programación usar para realizar el desarrollo. En este punto hay que recordar los 2 requisitos que se impusieron: el lenguaje debe ser *open source*, y debe tener a su vez librerías *open source* en las que apoyarnos para el uso de los algoritmos que vamos a probar.

Lenguaje de programación

Tras una búsqueda preliminar con esas 2 premisas el número de lenguajes fue reducido, principalmente a 2: Python y R. Ambos cumplen los 2 requisitos, impuestos, pero se ha escogido 1 de ellos y este ha sido Python.

Ilustración 8: Python logo

Las razones que han hecho que se use este lenguaje han sido las siguientes:

- Es un lenguaje interpretado, rápido y con una sintaxis clara y sencilla.
- Es uno de los lenguajes que más ha crecido en estos últimos años en su uso para inteligencia artificial [14].
- Las herramientas Big Data más usadas tienen sus *APIs* escritas en Python (*Apache Spark*).
- Posee un gran abanico de librerías *Open Source* para el desarrollo de proyectos en inteligencia artificial.
- Además, Python cuenta con una extensa colección de librerías que ayudan en la manipulación de datos para realizar transformaciones y operaciones sobre conjuntos de datos grandes.

Por estas razones se ha escogido dicho lenguaje de programación para realizar la implementación del sistema.

Librería de aprendizaje automático

Una vez seleccionado el lenguaje, otra de las decisiones que se ha tenido que tomar, ha sido que librería de Python para aprendizaje automático escoger. Existen varias librerías para este lenguaje como *TensorFlow, Theano, Keras o Sci-kit learn*.

La librería que se escoja tiene que cumplir el requisito impuesto, que sea *Open Source* y además que tenga integrados los algoritmos que se van a utilizar en este estudio, para simplificar el tiempo de desarrollo.

De las librerías citadas anteriormente se han descartado las 3 primeras por los siguientes motivos:

- **TensorFlow:** Esta librería ha sido descartada por qué no contiene los algoritmos de máquinas de vectores de soporte para realizar regresiones. Además, su curva de aprendizaje es bastante alta.
- **Theano:** Esta librería es de muy bajo nivel, sirve para cualquier propósito, pero hay que desarrollar los algoritmos, por lo que queda descartada. Su curva de aprendizaje es la más pronunciada de todas.
- **Keras:** Es una *API,* que funciona sobre las 2 librerías citadas anteriormente, proporcionando cierto nivel de abstracción. Queda descartada por el mismo motivo que las 2 anteriores.

Esto deja como opción la última librería citada *Sci-Kit Learn* [18]. Esta librería cumple con los requisitos pedidos para este desarrollo. Además, su curva de aprendizaje es bastante suave, por lo que es fácil ponerse a trabajar con ella. Se puede obtener más información acerca de esta librería en [15].

Control de versiones

Para poder gestionar el código, así como las diferentes versiones de este producidas durante el desarrollo se decidió utilizar un gestor de código. La herramienta elegida fue *Git.* Esto ha supuesto ciertas ventajas a la hora de llevar acabo el desarrollo:

- Tener un repositorio central que permite tener copias de seguridad.
- Poder trabajar desde distintos equipos accediendo al repositorio remoto.
- Probar nuevas funcionalidades sin que afecte al código principal.
- Tener de manera automática un registro de los cambios realizados sobre el código.

Descripción de los datos

En este apartado vamos a describir el conjunto de datos de partida para el estudio, y así darle significado a cada uno de los campos para su mejor entendimiento.

En primer lugar, el conjunto de datos es un CSV, compuesto por 98241 lecturas recogidas a lo largo de 2 años, de 2015 a 2017. Cada lectura está compuesta por 10 campos que pasaremos a explicar a continuación:

- **Índice:** Es el primer campo, y es tan solo un contador para enumerar las filas que tiene el conjunto de datos. No tiene ninguna relevancia.
- **Fecha:** Este campo se refiera a la fecha y hora en la que se ha realizado la lectura. Utiliza un formato similar al estándar ISO 8601, usando el formato YYYY-MM-DD hh:mm:ss, donde YYYY corresponde al año, MM corresponde al mes, DD corresponde al día, hh corresponde a la hora, mm corresponde a los minutos y ss corresponde a los segundos.
- **Fecha Prevista:** Este campo hace referencia a la fecha y hora de la predicción. Utiliza el mismo formato que el campo anterior. Este campo es el cálculo de añadir el campo horizonte al campo fecha.
- **ghiCs Prevista:** Este campo hace referencia a la lectura de la máxima radiación solar registrada para el instante de tiempo referido en el campo date. Su valor es expresado con un número entero y una aproximación de 11 decimales.
- **Horizonte:** Este campo representa el horizonte de tiempo a predecir. Este valor tiene un dominio de [0,360] y esta expresado en minutos.

Los campos que vienen descritos a continuación hacen todos referencia a la predicción realizada para el instante de tiempo expresado en el campo fecha y el horizonte de tiempo expresado en el campo horizonte. Estas lecturas están expresadas en índice denominado *kt*. Este índice se obtiene dividiendo la predicción del campo entre el valor correspondiente del campo ghiCs Prevista. Se utilizan estas lecturas con el índice *kt*, principalmente, porque elimina el sesgo que introduce la hora del día. El índice *kt* se corresponde con la siguiente fórmula (Eq. 7):

$$ghiKT = \frac{GHI}{GHI_{cs}} \tag{7}$$

Ecuación 7: Fórmula del índice kt

- **ghiKt_CIADCast:** Este campo hace referencia a la predicción obtenida por el modelo físico *Cloud Index Advection and Diffusion*.
- **ghiKt_Satelite:** Este campo hace referencia a la predicción obtenida por el modelo físico *Satellite*.
- **ghiKt_SmartPersistence:** Este campo hace referencia a la predicción obtenida por el modelo físico de *Smart Persistence*.
- **ghiKt_WRFsolar:** Este campo hace referencia a la predicción obtenida por el modelo físico *Solar Weather Research and Forecasting*.

El último campo descrito es que hace referencia a la medida esperada de radiación solar para el instante de tiempo expresado en el campo date y el horizonte de tiempo expresado en el campo horizonte.

- **ghiKt Esperado:** Este campo representa la lectura esperada para la radiación solar.

Descripción del desarrollo

En este apartado se irá describiendo paso a paso todo el procedimiento seguido para alcanzar los resultados y como ha sido realizado.

Carga de datos

En primer lugar, lo que se ha hecho ha sido cargar los datos en memoria para poder trabajar con ellos. Los datos estaban almacenados en un fichero en formato CSV, y se han cargado en una estructura llamada *DataFrame*. Un *DataFrame* es una estructura de datos que nos permite acceder a ellos y representarlos como si fuesen una tabla, con esta representación se puede acceder a ellos a nivel de fila o columna, hacer consultas, u operaciones complejas.

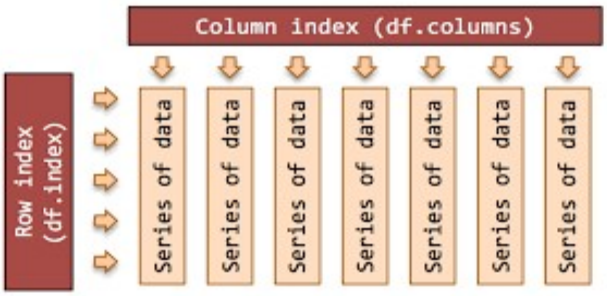

Ilustración 9: Ejemplo Dataframe

Esta estructura la se puede utilizar gracias a una librería *Open Source* llamada Pandas. Esta librería además nos provee de muchas funciones para poder manipular e interaccionar con este tipo de estructuras.

Normalización de los datos

Algunos algoritmos de aprendizaje automático requieren que los datos estén normalizados en rangos entre [0,1] o [-1,1] para poder funcionar correctamente. La normalización nos evita problemas derivados de la diferencia de magnitudes de las variables, en caso de que las hubiese, y también puede ayudar a mejorar el rendimiento de los algoritmos. Para evitar los posibles problemas, se han normalizado las columnas correspondientes a los modelos físicos, y la columna correspondiente a la predicción esperada.

Para llevar a cabo la normalización se ha utilizado la normalización basada en el máximo y el mínimo. Este proceso lo que hace es a cada valor restarle el valor mínimo en esa columna y después se divide entre la diferencia del máximo valor de la columna menos el mínimo valor de la columna, como se puede ver en la fórmula (Eq. 8).

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{8}$$

Ecuación 8: Normalización usando mínimo y máximo de un conjunto de datos

Una cosa importante al realizar esta normalización es almacenar los valores máximo y mínimo de cada columna para en un paso posterior deshacer esta operación y poder tener los valores en su magnitud original.

Transformación del conjunto de datos

La siguiente tarea que realizar es crear los 2 conjuntos de entrada diferentes que necesitamos, uno para el modelo general y otro para el modelo por vecindad.

Modelo general

El modelo general consiste, como ya vimos anteriormente, en usar los valores de cada uno de los cuatro modelos físicos para realizar una predicción. La fórmula general quedó reflejada anteriormente en la ecuación 2 (Eq 2).

Este modelo apenas requiere transformaciones ya que solo necesitamos quedarnos con las 4 columnas correspondientes a los predictores.

Modelo por vecindad

El modelo por vecindad, como ya se presentó anteriormente, consiste en usar los valores del horizonte anterior y posterior de uno de los modelos físicos para complementar los datos, es decir la fórmula de la predicción es la reflejada en la ecuación 4 (Eq. 4).

Esto nos lleva a la conclusión de que, si queremos predecir un valor, y no tiene datos en el horizonte anterior y posterior debemos eliminar ese registro del conjunto de datos, ya que no se puede predecir.

Para que los valores de los resultados sean consistentes, los valores en ambos modelos deben ser iguales, esto quiere decir que si al realizar la transformación del modelo de vecindad una fila ha sido eliminada por qué no cumplía los requisitos para estar presente, esa fila debe ser también eliminada en el modelo general.

Al finalizar las transformaciones, ambos modelos quedan con el mismo número de filas, teniendo ahora 4 columnas el modelo general, cada columna corresponde a 1 modelo físico, y 12 el modelo por vecindad, siendo 3 columnas pro cada uno de los modelos físicos, una para el valor del horizonte anterior, otra para el valor actual, y otra para el valor del horizonte posterior.

Validación cruzada

Una vez finalizados los pasos anteriores se procede a aplicar la validación cruzada, la explicación se dividirá en los pasos que se han seguido para poder realizarla.

División del conjunto de datos

Para poder aplicar la validación cruzada primero tenemos que dividir nuestro conjunto de datos en diferentes subconjuntos. Como se necesitaba una representación homogénea de los datos en los conjuntos de entrenamiento y prueba se ha optado por la siguiente aproximación.

Se ha decidido agrupar en subconjuntos de 4 días, es decir, se usarán 3 días para entrenar, y 1 para hacer pruebas sobre el modelo. La decisión se debe a que, para tener la mejor representación posible, al ser una serie temporal, es más probable que los días contiguos entre sí tengan unas condiciones climáticas similares, por ello se ha decidido hacer de esta manera.

Esto nos deja con 4 subconjuntos, que están divididos en 2 partes, una para entrenar y otra para hacer pruebas. La ilustración 4 sería un ejemplo de cómo quedarían divididos los conjuntos.

Ajuste de parámetros de los algoritmos

Después de haber dividido el conjunto de datos, antes de hacer la fase de entrenamiento, debemos encontrar los mejores valores para los hiperparámetros de los algoritmos. Algunos algoritmos de aprendizaje automático tienen ciertos valores, en forma de parámetros, que se pueden modificar, y pueden afectar a la calidad de las predicciones. Por ejemplo, en el caso del *Random Forest*, se puede ajustar el número de árboles/nodos que tiene el bosque.

Para que el ajuste de parámetros no afecte a la predicción final, en lugar de hacer las pruebas con los subconjuntos de pruebas, el subconjunto de entrenamiento se dividirá en 2, uno de entrenamiento y otro de validación. Con esto intentamos evitar que se pueda producir un sobreajuste en las predicciones finales, que alterarían los resultados.

De los algoritmos elegidos para este estudio solo 2 de ellos requieren de un ajuste de hiperparámetros, el algoritmo *Random Forest*, y la máquina de soporte de vectores con *kernel* radial. En el caso del *Random Forest* se ajustarán el número de árboles/nodos en el bosque, probaremos con diferentes valores desde 25, hasta 500, haciendo un incremento de 25 en 25 (25, 50, 75, etc). Para la máquina de soporte de vectores con *kernel* radial modificaremos el parámetro *gamma*, y probaremos los valores 0.001, 0.01, 0.1, 1, 0.0001.

Aplicar la validación cruzada

Una vez realizados los pasos anteriores, es el momento de poder aplicar la técnica de validación cruzada. Para cada uno de los cuatro subconjuntos creados, se procederá a entrenar el algoritmo correspondiente, usando los valores de entrenamiento, y aplicándole, en caso de que proceda, el hiperparámetro obtenido en la fase de ajuste. Una vez entrenado, se usarán los datos del subconjunto de prueba para probar el modelo.

Este proceso se realizará 4 veces, una por cada subconjunto, y se obtendrán unas medidas de errores en las predicciones (se comentarán en el siguiente punto). Para calcular el resultado final se hará la media de las cuatro ejecuciones y este será el resultado final.

Obtención de resultados

Por último, se extraen los resultados, y se almacenan en un fichero de texto en formato CSV, para poder ser interpretados más tarde. Ahora se explicará cómo se obtienen los resultados y que métricas se han utilizado para este proyecto.

Destacar que antes de poder realizar los cálculos de los errores, debemos devolverlos a su magnitud original, para ello, usando los valores almacenados antes para cada columna, procedemos a realizar la operación inversa y se desnormalizarán los valores.

Se evalúan cuatro métricas de error:

- **nMAE:** Esta métrica corresponde al error absoluto medio (por sus siglas en inglés MAE) normalizado. Este es calculado con la siguiente fórmula (Eq. 9), donde p_i corresponde a las predicciones realizadas, o_i a las observaciones (las predicciones esperadas).

$$nMAE = \frac{\sum |p_i - o_i|}{\sum o_i} \quad (9)$$

Ecuación 9: Error absoluto medio normalizado

- **nRMSE:** Esta métrica corresponde al error cuadrático medio (por sus siglas en inglés RMSE) normalizado. Este es calculado con la siguiente fórmula (Eq. 10), donde p_i corresponde a las predicciones realizadas, o_i a las observaciones (las predicciones esperadas) y N representa el número de muestras en el conjunto de datos.

$$nRMSE = \frac{\sqrt{\sum (p_i - o_i)^2 \, / N}}{\sum (o_i)/N} \quad (10)$$

Ecuación 10: Error cuadrático medio normalizado

- **MAE:** Esta métrica corresponde al error absoluto medio (por sus siglas en inglés MAE). Este es calculado con la siguiente fórmula (Eq. 11), donde p_i corresponde a las predicciones realizadas, o_i a las observaciones (las predicciones esperadas) y N representa el número de muestras en el conjunto de datos

$$MAE = \frac{\sum |p_i - o_i|}{N} \quad (11)$$

Ecuación 11: Error absoluto medio

- **RMSE:** Esta métrica corresponde al error cuadrático medio (por sus siglas en inglés RMSE). Este es calculado con la siguiente fórmula (Eq. 12), donde p_i corresponde a las predicciones realizadas, o_i a las observaciones (las predicciones esperadas) y N representa el número de muestras en el conjunto de datos.

$$RMSE = \sqrt{\frac{\sum (p_i - o_i)^2}{N}} \quad (12)$$

Ecuación 12: Error cuadrático medio

Estas cuatro métricas serán calculadas, en ambos modelos, por cada horizonte, y el error global de cada métrica será la media entre el valor de cada uno de los horizontes. Las métricas han sido escogidas por su significado. El error cuadrático medio, RMSE, se ha incorporado porque se utiliza en el campo de la física para medir el error entre predicciones. El error absoluto medio, MAE, por otra parte, se incorporó porque se utiliza por motivos económicos en el mercado energético. En el siguiente capítulo se procederá a presentar los resultados obtenidos por este procedimiento.

Capítulo 5

Resultados

Resultados

A continuación, se presentarán los resultados obtenidos en el estudio. Después de cada resultado se comentará brevemente para ver que se ha obtenido, y como se interpretan esas métricas.

La batería de pruebas realizada ha sido la siguiente: Para cada uno de los algoritmos, 4 en total, se ha probado cada uno de los modelos con cada una de las aproximaciones, lo que hace un total de 16 pruebas.

Los resultados se expondrán de la siguiente manera: tenemos 2 modelos diferentes, uno general, o sin vecindad, y un modelo por vecindad, además tenemos 2 aproximaciones, una aproximación general, que consiste en que tenemos una única función de predicción global, y una aproximación por horizontes, en la cual, tenemos una función por cada horizonte a predecir.

Error global

En este apartado se verá el valor del error global de cada una de las métricas. Se dividirá en 2, un apartado para cada aproximación.

Aproximación general

Esta es la tabla con los valores de la aproximación general.

Algorithm	Architecture	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	0.1540888497	0.2475945287	81.4542455790	131.3990789257
Linear Regression	Vecindad	0.1530237866	0.2459126542	80.8467740350	130.4724214863
SVM Linear Kernel	Normal	0.1485030689	0.2560045466	78.4070186130	135.9339166109
SVM Linear Kernel	Vecindad	0.1486037612	0.2541462914	78.4418828382	134.9246803849
SVM Radial Kernel	Normal	0.1629278371	0.2488679504	86.3060786944	132.1515952640
SVM Radial Kernel	Vecindad	0.1589810816	0.2469360821	84.1034742718	131.0646519330
Random Forest	Normal	0.1568198169	0.2577160418	82.7929812553	136.7430598100
Random Forest	Vecindad	0.1514668469	0.2499372661	79.9736218161	132.6292007967

Tabla 18: Errores por algoritmo de la aproximación general

Como se puede observar en la tabla, en la aproximación general, el modelo por vecindad es mejor en todos los casos de forma consistente, aunque la mejora es pequeña, presenta mejores resultados. En el caso del algoritmo SVM con *kernel* lineal, aunque se vea que el modelo por vecindad es marginalmente peor en los resultados en nMAE, se puede observar que es mejor en el nRMSE, y esto se debe a que aun que la media de los errores sea ligeramente menor, el nRMSE penaliza en mayor medida los errores de mayor magnitud, por lo que para este modelo es más interesante tener en cuenta en mayor medida esta métrica de error.

Aproximación por horizontes

Esta es la tabla con los valores de la aproximación por horizontes.

Algorithm	Architecture	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	0.15541363101	0.24932568038	81.92981942682	132.07962010413
Linear Regression	Vecindad	0.15426508400	0.24804070165	81.31179798785	131.36296184847
SVM Linear Kernel	Normal	0.14936152820	0.25721694664	78.72348697905	136.41575522645
SVM Linear Kernel	Vecindad	0.14951601472	0.25539023206	78.80810541496	135.43498757066
SVM Radial Kernel	Normal	0.17024153757	0.25162711833	89.77536706860	133.32436783367
SVM Radial Kernel	Vecindad	0.16449212257	0.24889992425	86.69939911595	131.86697549556
Random Forest	Normal	0.15591017901	0.25833259094	82.07842745842	136.76619638001
Random Forest	Vecindad	0.15280297174	0.25210936747	80.49557454683	133.59773813678

Tabla 19: Errores por algoritmo de la aproximación por Horizontes

En esta tabla se puede observar, que como en la aproximación general, el modelo por vecindad mejora respecto al modelo general. En caso del algoritmo SVM con *kernel* lineal observamos el mismo caso que en el ejemplo anterior. Aunque la media de los errores es menor, la distancia entre la predicción y la observación es mayor, por lo que se puede asegurar que el modelo por vecindad presenta mejores resultados.

Error por horizonte

En este apartado se mostrarán los errores por horizonte. Se presentarán los resultados por algoritmo, y dentro de esa división se presentarán para cada una de las métricas. En cada gráfica se presentarán las 2 aproximaciones usadas, la general y la de por horizontes, para poder compararlas entre sí y reflejar mejor el resultado.

Regresión lineal

El primer algoritmo presentado será la regresión lineal simple, a continuación, estas son sus métricas de errores.

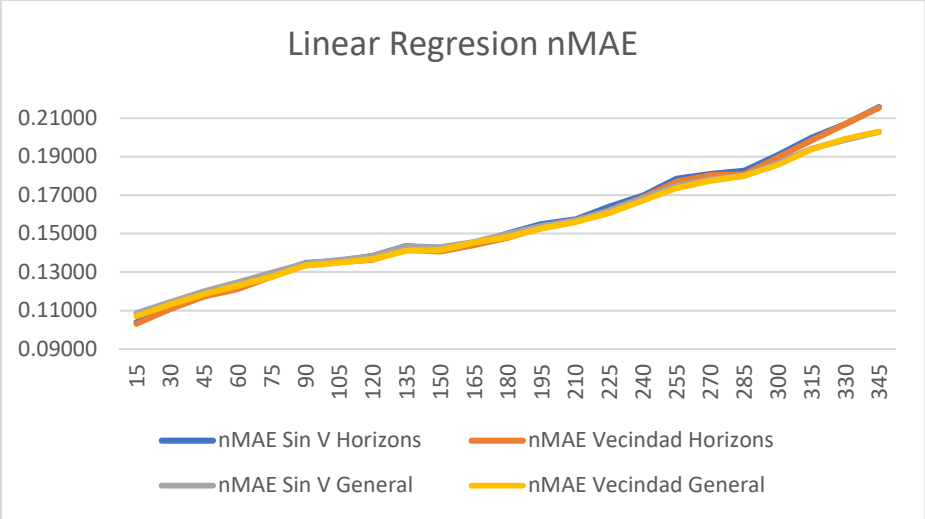

Ilustración 10: Gráfica de la métrica nMAE para el algoritmo de regresión lineal

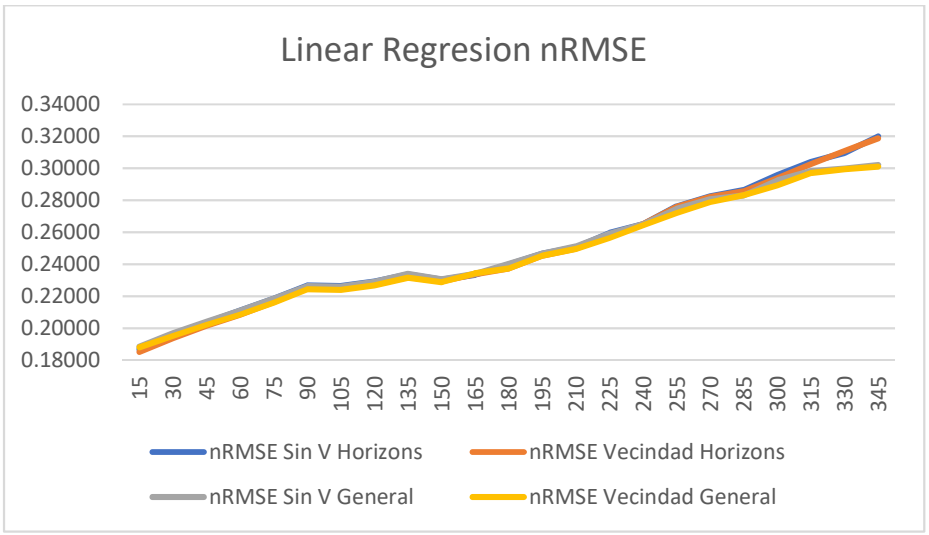

Ilustración 11: Gráfica de la métrica nRMSE para el algoritmo de regresión lineal

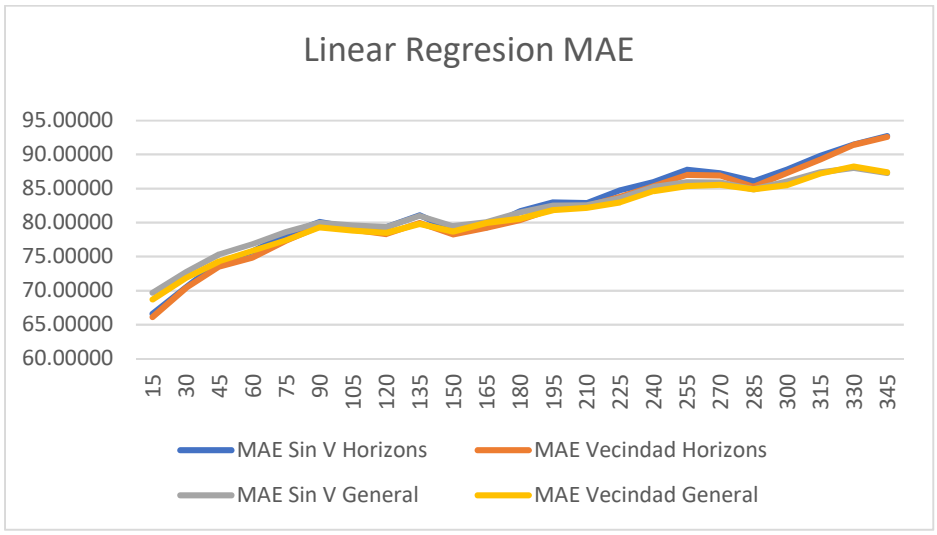

Ilustración 12: Gráfica de la métrica MAE para el algoritmo de regresión lineal

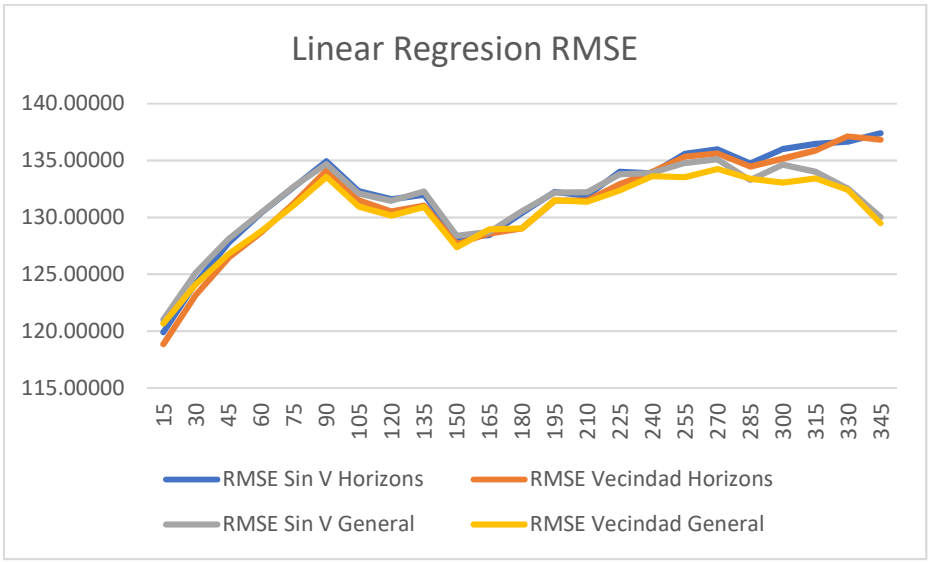

Ilustración 13: Gráfica de la métrica RMSE para el algoritmo de regresión lineal

Como se puede observar en la ilustración 10, el modelo por vecindad presenta mejores resultados en horizontes cercanos, siendo, en ambas aproximaciones, mejor que el modelo normal. También se puede ver que la arquitectura por horizontes obtiene mejores resultados cuanto más cerca del momento de la predicción, pero en horizontes lejanos no es tan bueno, sin embargo, la arquitectura general es algo mejor en horizontes lejanos. Se puede confirmar esta tendencia también en la ilustración 11.

Se puede apreciar mejor en las ilustraciones 12 y 13 lo comentado anteriormente, y se puede observar como el error va aumentando con el horizonte de forma gradual.

SVM con kernel lineal

El siguiente algoritmo que presentar es la máquina de soporte de vectores con *kernel* lineal. Estos son los resultados de su métrica de errores.

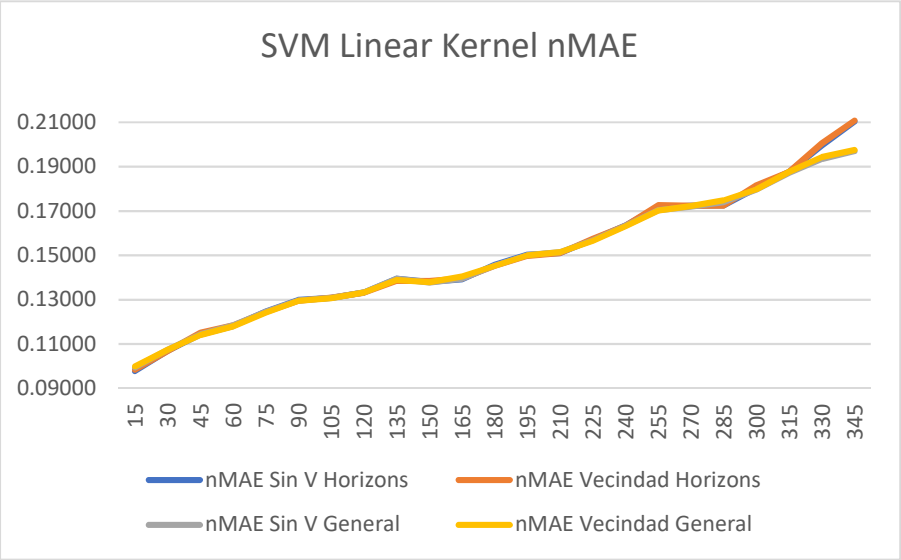

Ilustración 14: Gráfica de la métrica nMAE para el algoritmo SVM Lineal

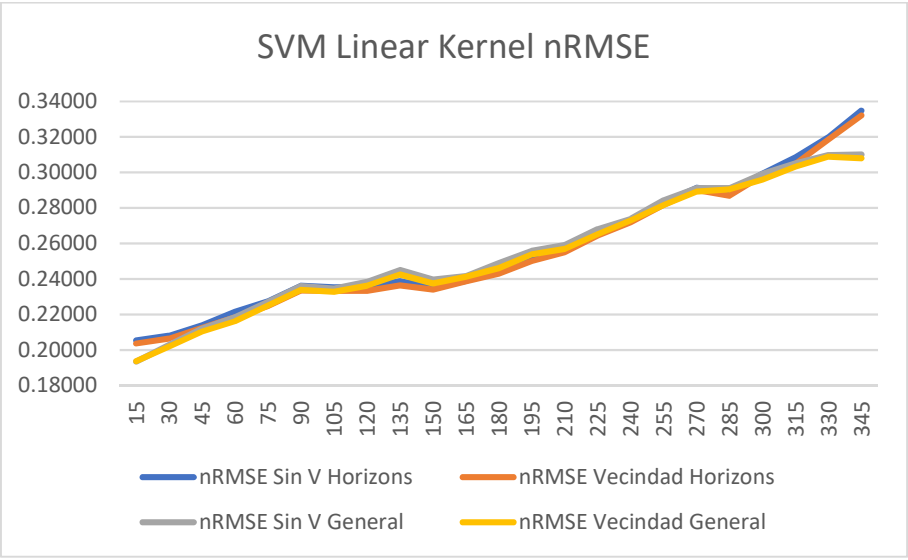

Ilustración 15: Gráfica de la métrica nRSME para el algoritmo SVM Lineal

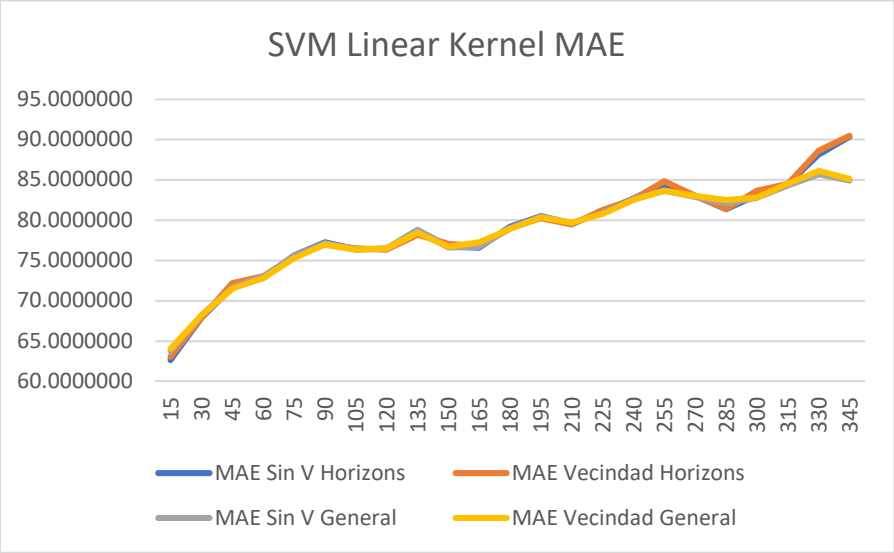

Ilustración 16: Gráfica de la métrica MAE para el algoritmo SVM Lineal

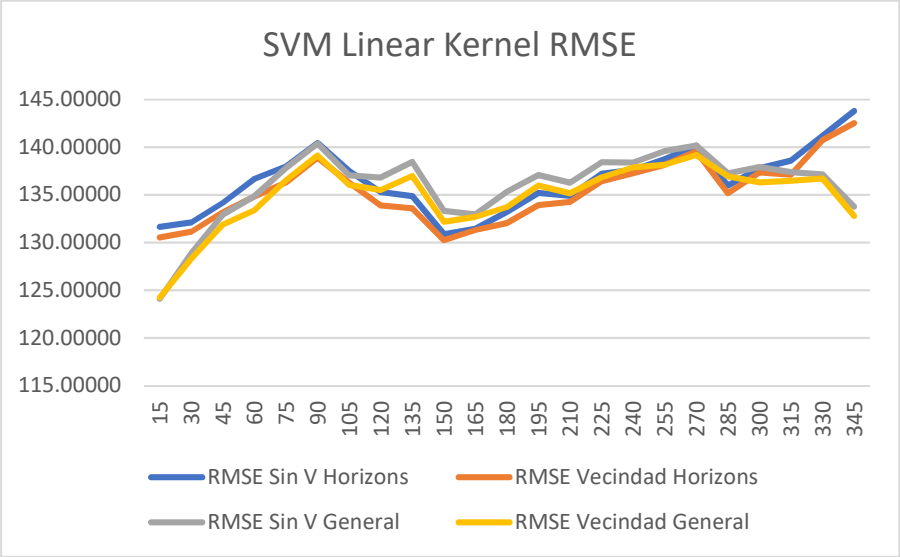

Ilustración 17: Gráfica de la métrica RSME para el algoritmo SVM Lineal

En la ilustración 14, no se pude apreciar (se pueden observar con más detalle en el Anexo B), pero en los datos muestra un mejor resultado el modelo normal, siendo unas milésimas mejor, pero si se hace un estudio de la otra métrica, como se ve en la ilustración 15, se puede observar que el modelo por vecindad obtiene mejores resultados. Esto puede explicarse por qué, aun siendo menor la media del error en el modelo general, existen errores con mayor distancia de la predicción, que es lo que penaliza la segunda métrica, por eso en este caso el modelo por vecindad sigue presentando mejores resultados. También se puede apreciar una tendencia similar a la del ejemplo anterior, aunque en este caso la aproximación general muestra mejores resultados que la aproximación por horizontes.

En las ilustraciones 16 y 17 se puede observar lo comentado anteriormente.

SVM con kernel Radial

A continuación, se presenta el algoritmo máquina de soporte de vectores con *kernel* radial. Estos son los resultados de su métrica de errores.

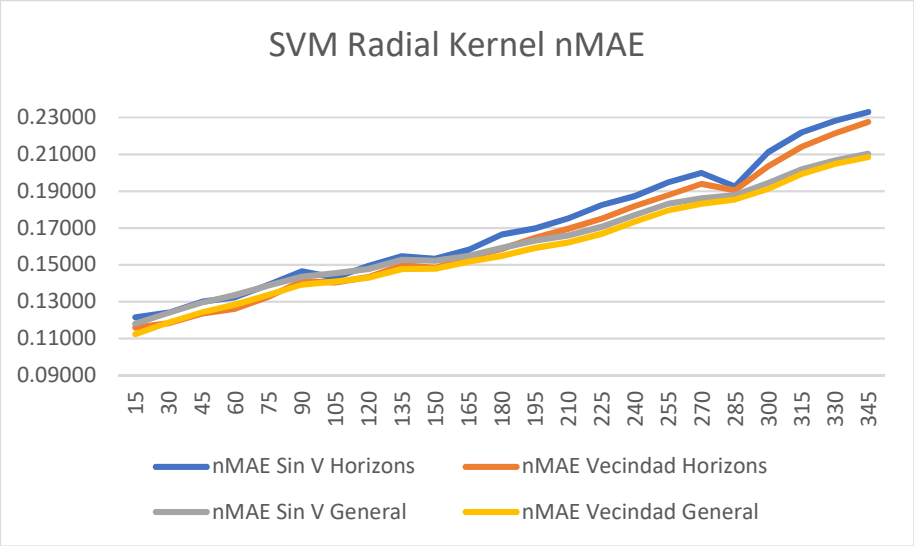

Ilustración 18: Gráfica de la métrica nMAE para el algoritmo SVM Radial

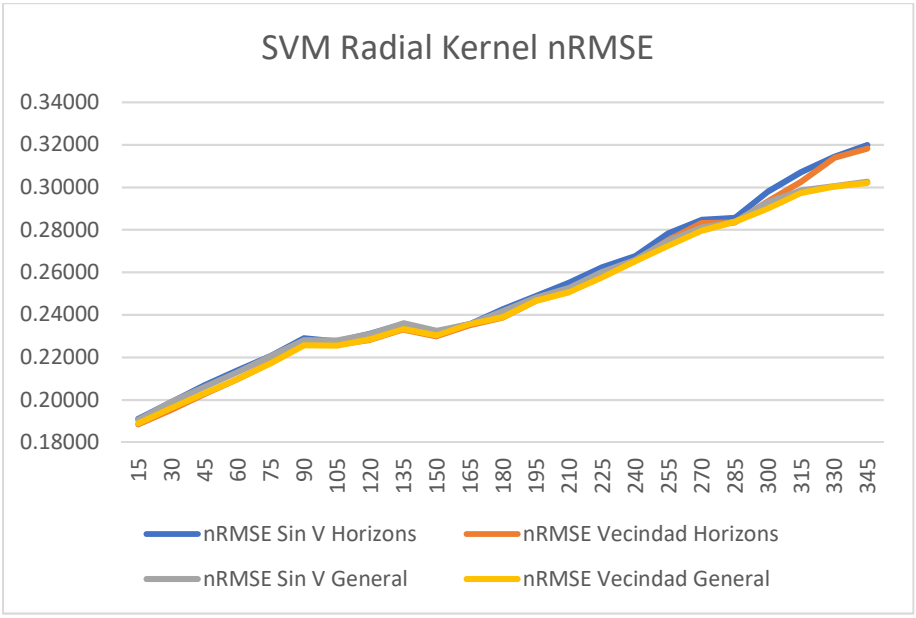

Ilustración 19: Gráfica de la métrica nRMSE para el algoritmo SVM Radial

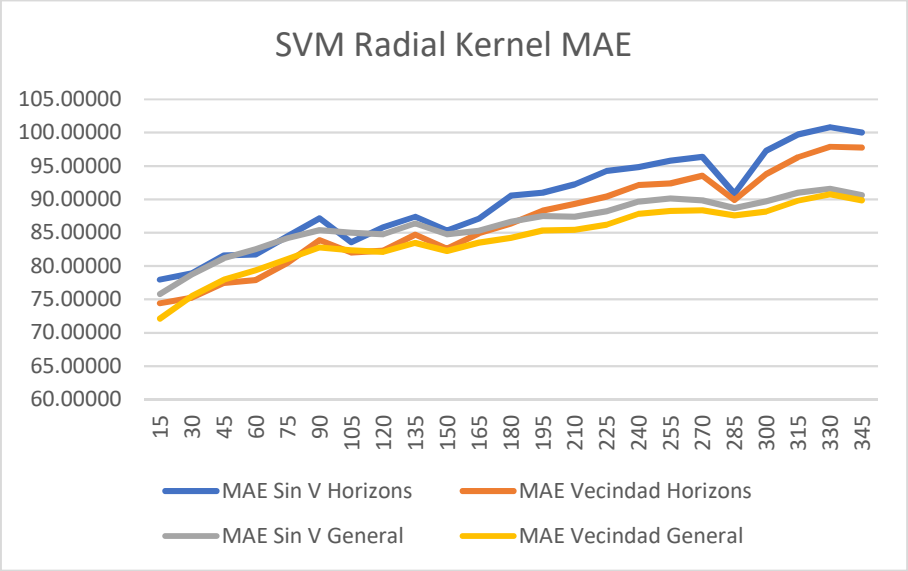

Ilustración 20: Gráfica de la métrica MAE para el algoritmo SVM Radial

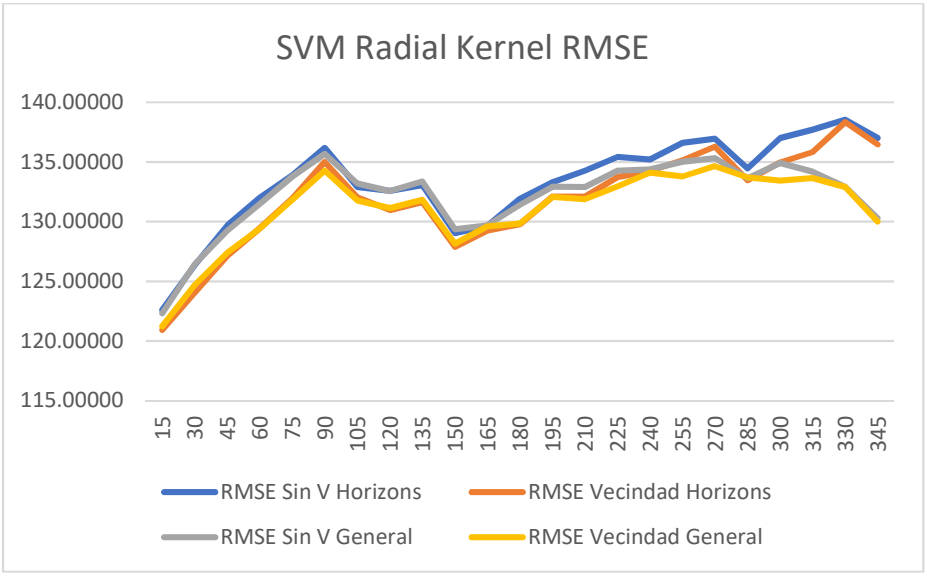

Ilustración 21: Gráfica de la métrica RMSE para el algoritmo SVM Radial

En este nuevo ejemplo, como se observa en la ilustración 18, los modelos por vecindad muestran mejores resultados que los modelos generales. Como ocurría en el ejemplo anterior la aproximación general muestra mejores resultados en la primera gráfica, pero si analizamos la ilustración 19, se puede observar que la aproximación por horizontes muestra mejores resultados en los horizontes más cercanos, y conforme nos alejamos en el tiempo, la aproximación general muestra mejores resultados.

En las ilustraciones 20 y 21 se puede apreciar mejor los explicado anteriormente.

Random Forest

Por último, se presentan los resultados del último algoritmo de ejemplo. Estos son sus resultados.

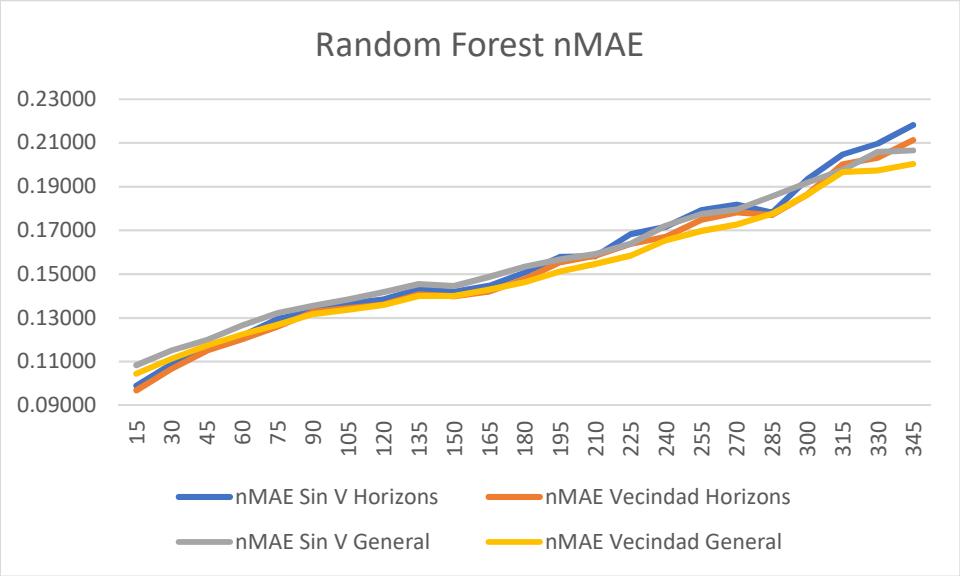

Ilustración 22: Gráfica de la métrica nMAE para el algoritmo Random Forest

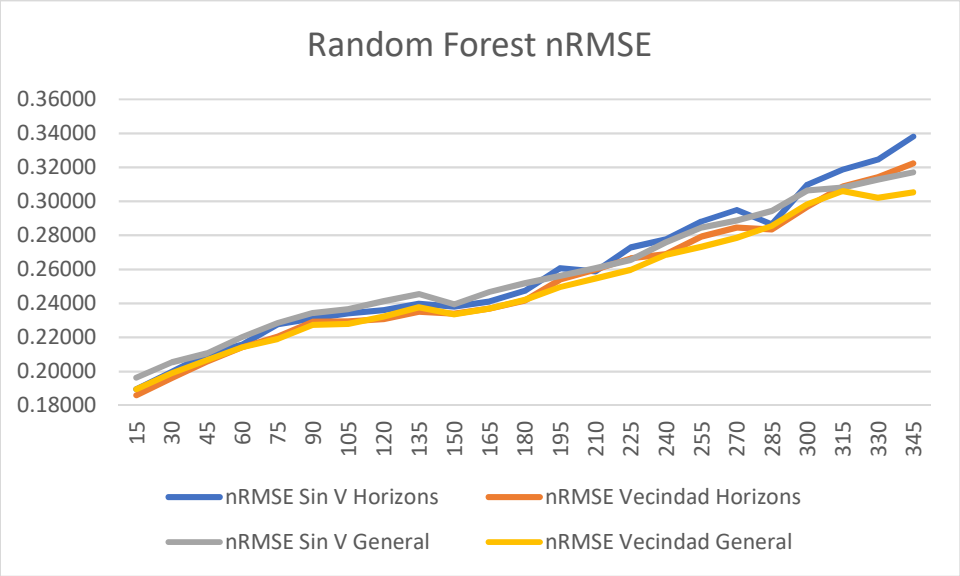

Ilustración 23: Gráfica de la métrica nRMSE para el algoritmo Random Forest

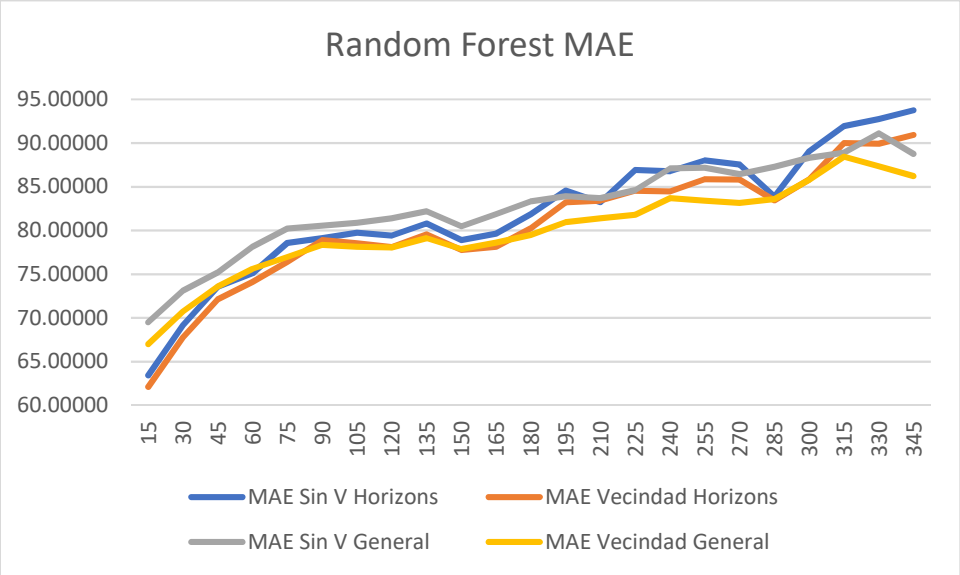

Ilustración 24: Gráfica de la métrica MAE para el algoritmo Random Forest

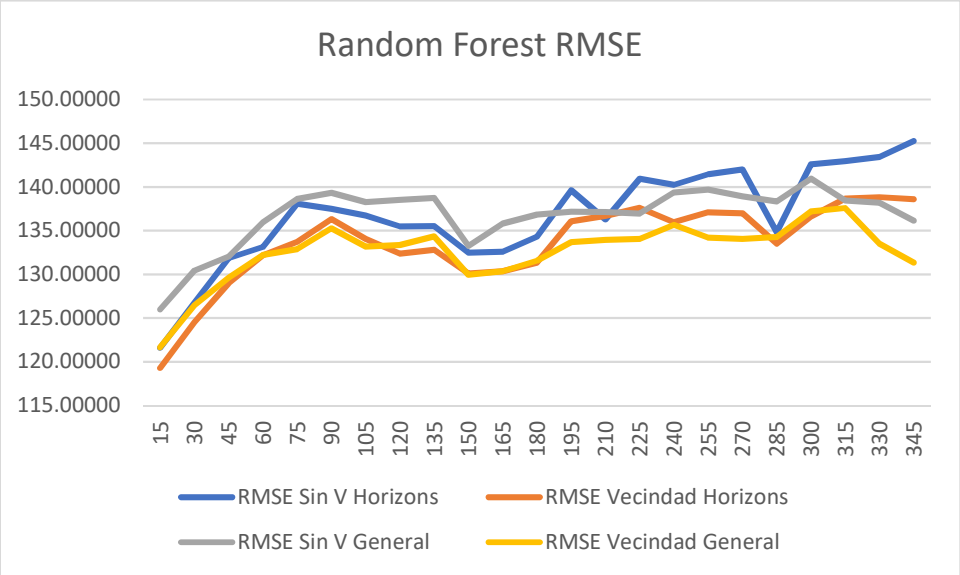

Ilustración 25: Gráfica de la métrica RMSE para el algoritmo Random Forest

En este último ejemplo, como se observa en la ilustración 22, se puede ver que el modelo por vecindad obtiene mejores resultados en ambas aproximaciones, siendo ligeramente mejor la aproximación por horizontes. Se puede observar en la ilustración 23 como en este ejemplo el error se incrementa según se alejan en el tiempo, pero ambas aproximaciones están muy parejas, donde normalmente empezaba a empeorar la aproximación por horizontes, esta vez la diferencia es ligeramente menor en horizontes lejanos.

En las ilustraciones 24 y 25 se pueden observar las tendencias descritas anteriormente.

Capítulo 6

Conclusiones y líneas futuras

Conclusiones

Tras mostrar los resultados y ser analizados se puede concluir que de forma sistemática el modelo por vecindad muestra mejores resultados. Esto quiere decir que mejora el modelo general usado hasta ahora, lo que supone un resultado satisfactorio para el estudio realizado.

En cuanto a las 2 aproximaciones utilizadas, una general y otra por horizontes, se puede ver como la aproximación por horizontes suele resultar mejor a la hora de predecir valores en horizontes medios y próximos, mientras que la aproximación general obtiene mejores resultados en horizontes lejanos.

Que la aproximación por horizontes tenga peores resultados en horizontes lejanos tiene una explicación. Las predicciones de los modelos físicos están hechas hasta 6 horas, esto quiere decir que, por ejemplo, un día normal de invierno oscurecerá sobre las 6 P.M de la tarde, esto implica que las lecturas hechas a partir de las 12 P.M empezaran a perder gradualmente información al no contar con los horizontes finales. Esto implica que los horizontes finales suelen tener menos datos, por lo que las predicciones tenderán a ser menos exactas con esta aproximación. La aproximación general es capaz de compensar esto con todos los datos, pero el modelo por horizontes, no.

Con esto se puede concluir que existe una mejora con respecto a las aproximaciones actuales y que hace que merezca la pena darle una oportunidad a esta aproximación. Aunque la mejora ha sido marginal esto no implica que los resultados sean malos, si no que necesitemos hacer un estudio más profundo sobre el modelo por vecindad.

Con respecto a los objetivos planteados, se han cumplido todos, ya que de no haber sido así, no se hubiese podido finalizar este proyecto. La muestra de resultados es la culminación de la consecución de los objetivos marcados.

Opinión personal

Quería destacar también que, gracias a la realización de este proyecto, mi conocimiento personal se ha visto gratamente incrementado. He podido aprender un nuevo lenguaje de programación, Python, que apenas había usado alguna vez para hacer pequeños ejemplos. Mis conocimientos en técnicas de aprendizaje automático, así como su funcionamiento también ha mejorado, ya que normalmente no se tienen oportunidades de aplicar estos conocimientos a un caso práctico real como este. También me ha servido para recordar conocimientos adquiridos durante mi carrera que hasta ahora no había tenido oportunidad de emplear. Por todo esto, y también gracias a los resultados obtenidos, mi opinión es que ha merecido mucho la pena desarrollar este proyecto de investigación.

Líneas futuras

Como apunte final veremos algunas de las líneas con las que se podría ampliar este estudio. En primer lugar, se podría añadir algún otro modelo de aprendizaje automático, y así compararlo con estos resultados, para poder tener un abanico más amplio de posibilidades. Como se ha comentado en las conclusiones, un estudio más profundo del modelo de vecindad, por ejemplo, intentar hacer un modelo de vecindad expandido, con más horizontes, o un modelo que utilice todos los horizontes para hacer una predicción.

Otra idea sería probar más algoritmos, para poder ver el comportamiento y ampliar los datos del estudio. También se podrían añadir más modelos físicos o utilizar otros diferentes.

Por último, una línea interesante, sería poder contar con un mayor número de datos para trabajar. Por qué cada día completo cuenta como un solo ejemplo. Un día puede ser nuboso, soleado o mixto, nunca sale de esos 3 casos, entonces se podría considerar que todos los datos de un día son el mismo. Por lo que sería muy interesante incluir más días, y así tener más muestras que usar para entrenar el modelo, y quizás así se podrían mejorar los resultados. También se podría contar con otros datos diferentes, por ejemplo, incluir datos meteorológicos, para añadir como entradas en el aprendizaje.

Anexo A- Presupuesto y Planificación

En este anexo se presentará la planificación realizada para este proyecto. También se presentará una relación de costes en un supuesto que este estudio hubiese sido contratado por un cliente.

Planificación

Este proyecto se ha dividido en 4 etapas: Investigación, Análisis, Desarrollo y Documentación. Cada una de estas etapas esta a su vez dividida en diferentes tareas que se han ido llevando a cabo durante el desarrollo del proyecto. La duración del proyecto está estimada en 300 horas, que han sido divididas en jornadas de 4 horas, lo que nos da un total de 75 días.

En la siguiente tabla se presenta como quedaría estructurado:

Planificación	
Etapa 1	Investigación
Tarea 1.1	Recolección de fuentes sobre el problema
Tarea 1.2	Organización de las fuentes
Etapa 2	Análisis
Tarea 2.1	Análisis del problema
Tarea 2.2	Estudio de las técnicas de aprendizaje automático
Tarea 2.3	Definición de requisitos
Tarea 2.4	Toma de decisiones
Tarea 2.5	Definición de la batería de pruebas
Etapa 3	Desarrollo
Tarea 3.1	Implementación del proyecto
Tarea 3.2	Experimentación
Etapa 4	Documentación
Tarea 4.1	Estudio de los resultados obtenidos
Tarea 4.2	Documentación del proyecto completo

Tabla 20: Etapas de planificación

A continuación, se muestra el proyecto en un diagrama de Gantt:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Estapa 1**	**10 días**	**vie 01/06/18**	**jue 14/06/18**
2	Tarea 1.1	8 días	vie 01/06/18	mar 12/06/18
3	Tarea 1.2	2 días	mié 13/06/18	jue 14/06/18
4	**Etapa 2**	**15 días**	**vie 15/06/18**	**jue 05/07/18**
5	Tarea 2.1	4 días	vie 15/06/18	mié 20/06/18
6	Tarea 2.2	4 días	jue 21/06/18	mar 26/06/18
7	Tarea 2.3	3 días	mié 27/06/18	vie 29/06/18
8	Tarea 2.4	2 días	lun 02/07/18	mar 03/07/18
9	Tarea 2.5	2 días	mié 04/07/18	jue 05/07/18
10	**Etapa 3**	**20 días**	**vie 06/07/18**	**jue 02/08/18**
11	Tarea 3.1	18 días	vie 06/07/18	mar 31/07/18
12	Tarea 3.2	2 días	mié 01/08/18	jue 02/08/18
13	**Estapa 4**	**30 días**	**vie 03/08/18**	**jue 13/09/18**
14	Tarea 4.1	10 días	vie 03/08/18	jue 16/08/18
15	Tarea 4.2	20 días	vie 17/08/18	jue 13/09/18

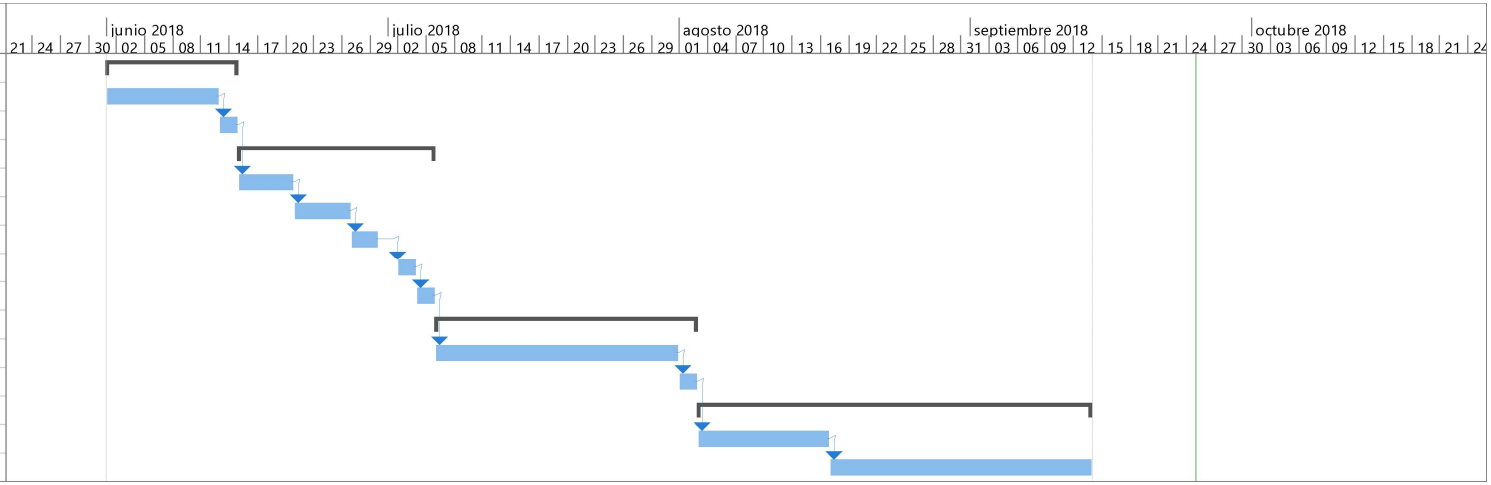

Ilustración 26: Planificación del proyecto. Diagrama de Gantt

Presupuesto

A continuación, se presenta la tabla de costes presupuestados. Este presupuesto corresponde a un supuesto en el que un posible cliente hubiese encargado realizar este proyecto. El precio estipulado para este proyecto por hora para un trabajador es de 21 euros la hora. Los días se contabilizarán como jornadas de 4 horas. Los costes serían los reflejados en la siguiente tabla:

Presupuesto				
Concepto	Precio	Vida útil	Uso Estimado	Precio final
Equipo para el desarrollo	1500€	60 meses	75 días	62€
Analista/ Programador	21€/hora	75 días	75 días	6300€

Tabla 21: Estimación de presupuesto

El coste total del proyecto ascendería a 6.362€. Como aclaración, el precio final del equipo para el desarrollo es el resultado de la amortización con respecto a su vida útil.

Anexo B- Resultados completos del proyecto

En el punto de resultados se explicó que los valores al ser tan reducidos en magnitud hay veces que cuesta apreciarlos en las gráficas. Por lo que en este anexo se van a incluir las tablas con los valores generados con la batería de pruebas y representados en las gráficas para si se quiere revisar con más detalle se tenga la oportunidad de hacerlo.

La estructura que presentan las tablas en es la siguiente. Cada tabla tiene 8 campos que explicamos a continuación:

- **Algoritmo:** Este campo indica el nombre del algoritmo que ha generado esa tabla.
- **Arquitectura:** Este campo se corresponde con el modelo. Puede tener 2 valores, normal o vecindad. Normal hace alusión al modelo general sin vecindad.
- **Aproximación:** Este campo tiene 2 valores, General u Horizontes. Cada uno de ellos se corresponde con las aproximaciones explicadas anteriormente.
- **Horizonte:** Este campo indica a que horizonte pertenece la predicción. Cuando este campo tiene el valor global, indica que es la media realizada con todos los horizontes.
- **nMAE:** Este campo indica la medición del error correspondiente al nombre del campo.
- **nRMSE:** Este campo indica la medición del error correspondiente al nombre del campo.
- **MAE:** Este campo indica la medición del error correspondiente al nombre del campo.
- **RMSE:** Este campo indica la medición del error correspondiente al nombre del campo.

A continuación, se presentarán las tablas de resultados. Se van a dividir en secciones según el algoritmo que las ha generado, para que se corresponda con los resultados expuestos.

Regresión lineal

Datos del modelo general con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	Horizons	15	0.1038577	0.1869574	66.6212026	119.8931672
Linear Regression	Normal	Horizons	30	0.1109757	0.1955892	70.5070468	124.2200267
Linear Regression	Normal	Horizons	45	0.1183108	0.2037736	74.1992382	127.7387079
Linear Regression	Normal	Horizons	60	0.1227345	0.2113134	75.7630134	130.3834899
Linear Regression	Normal	Horizons	75	0.1289291	0.2186790	78.2241501	132.6672306
Linear Regression	Normal	Horizons	90	0.1346615	0.2269429	80.0826847	134.9510343
Linear Regression	Normal	Horizons	105	0.1356385	0.2265234	79.2224519	132.3096415
Linear Regression	Normal	Horizons	120	0.1383112	0.2293540	79.3674493	131.5926518
Linear Regression	Normal	Horizons	135	0.1435041	0.2337171	81.0787493	132.0030015
Linear Regression	Normal	Horizons	150	0.1417622	0.2304844	78.8618988	128.1970572
Linear Regression	Normal	Horizons	165	0.1441473	0.2335156	79.3128921	128.4432583
Linear Regression	Normal	Horizons	180	0.1502159	0.2400118	81.6631486	130.3826158
Linear Regression	Normal	Horizons	195	0.1548385	0.2469385	82.9893213	132.2600104
Linear Regression	Normal	Horizons	210	0.1573806	0.2507188	82.8748857	131.9033032
Linear Regression	Normal	Horizons	225	0.1640448	0.2597284	84.7396653	134.0338196
Linear Regression	Normal	Horizons	240	0.1698445	0.2650768	85.9314450	133.8952357
Linear Regression	Normal	Horizons	255	0.1785070	0.2763394	87.7621467	135.6014777
Linear Regression	Normal	Horizons	270	0.1810512	0.2827348	87.2687015	136.0024096
Linear Regression	Normal	Horizons	285	0.1825946	0.2864487	86.0821196	134.7488800
Linear Regression	Normal	Horizons	300	0.1907497	0.2958585	87.8299797	136.0355169
Linear Regression	Normal	Horizons	315	0.1997431	0.3041159	89.8128519	136.4874792
Linear Regression	Normal	Horizons	330	0.2069559	0.3096967	91.4779682	136.6658480
Linear Regression	Normal	Horizons	345	0.2157551	0.3199723	92.7128360	137.4153993
Linear Regression	Normal	Horizons	Global	0.1554136	0.2493257	81.9298194	132.0796201

Tabla 22: Modelo general con la aproximación por horizontes para Regresión Lineal

Datos del modelo por vecindad con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Linear Regression	Vecindad	Horizons	15	0.1030771	0.1853192	66.1222113	118.8402927
Linear Regression	Vecindad	Horizons	30	0.1108387	0.1939522	70.4250207	123.1940690
Linear Regression	Vecindad	Horizons	45	0.1172698	0.2017226	73.5479975	126.4565345
Linear Regression	Vecindad	Horizons	60	0.1212815	0.2085052	74.8654718	128.6607366
Linear Regression	Vecindad	Horizons	75	0.1275533	0.2164034	77.3814120	131.2689190
Linear Regression	Vecindad	Horizons	90	0.1335720	0.2255519	79.4368348	134.1393061
Linear Regression	Vecindad	Horizons	105	0.1352085	0.2251847	78.9716032	131.5283301
Linear Regression	Vecindad	Horizons	120	0.1364330	0.2274860	78.2889171	130.5211778
Linear Regression	Vecindad	Horizons	135	0.1415131	0.2320220	79.9520705	131.0509772
Linear Regression	Vecindad	Horizons	150	0.1407148	0.2295252	78.2794338	127.6621583
Linear Regression	Vecindad	Horizons	165	0.1440186	0.2337882	79.2336196	128.5847450
Linear Regression	Vecindad	Horizons	180	0.1478676	0.2374602	80.3939212	129.0276798
Linear Regression	Vecindad	Horizons	195	0.1529878	0.2454636	82.0177958	131.4917549
Linear Regression	Vecindad	Horizons	210	0.1564685	0.2498699	82.4006034	131.4866234
Linear Regression	Vecindad	Horizons	225	0.1621629	0.2576415	83.7646216	132.9351459
Linear Regression	Vecindad	Horizons	240	0.1687686	0.2652737	85.3808865	134.0100324
Linear Regression	Vecindad	Horizons	255	0.1770262	0.2759648	86.9933845	135.3409126
Linear Regression	Vecindad	Horizons	270	0.1802909	0.2820015	86.9088435	135.6396112
Linear Regression	Vecindad	Horizons	285	0.1807573	0.2857014	85.2586591	134.4859662
Linear Regression	Vecindad	Horizons	300	0.1895049	0.2939152	87.2735092	135.1726868
Linear Regression	Vecindad	Horizons	315	0.1984631	0.3026690	89.2590358	135.8829591
Linear Regression	Vecindad	Horizons	330	0.2068676	0.3108002	91.4425880	137.1385770
Linear Regression	Vecindad	Horizons	345	0.2154513	0.3187144	92.5729128	136.8289270
Linear Regression	Vecindad	Horizons	Global	0.1542651	0.2480407	81.3117980	131.3629618

Tabla 23: Modelo por vecindad con la aproximación por horizontes para Regresión Lineal

Datos del modelo general con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	General	15	0.108632	0.188677	69.688357	121.025362
Linear Regression	Normal	General	30	0.114437	0.197015	72.71125	125.1551
Linear Regression	Normal	General	45	0.120143	0.204326	75.346804	128.099682
Linear Regression	Normal	General	60	0.124512	0.211324	76.86234	130.406076
Linear Regression	Normal	General	75	0.129583	0.218778	78.624913	132.734577
Linear Regression	Normal	General	90	0.1344	0.226518	79.932375	134.704823
Linear Regression	Normal	General	105	0.136226	0.22617	79.575294	132.116136
Linear Regression	Normal	General	120	0.138232	0.229067	79.332821	131.449263
Linear Regression	Normal	General	135	0.143365	0.234148	81.012557	132.27091
Linear Regression	Normal	General	150	0.142861	0.230751	79.488651	128.38097
Linear Regression	Normal	General	165	0.145464	0.234016	80.04831	128.740012
Linear Regression	Normal	General	180	0.149852	0.240322	81.466223	130.56469
Linear Regression	Normal	General	195	0.153922	0.246832	82.499441	132.204106
Linear Regression	Normal	General	210	0.156819	0.25133	82.560839	132.208598
Linear Regression	Normal	General	225	0.161764	0.259189	83.578766	133.784833
Linear Regression	Normal	General	240	0.168507	0.265058	85.242648	133.921142
Linear Regression	Normal	General	255	0.174774	0.274591	85.928135	134.775667
Linear Regression	Normal	General	270	0.178143	0.280737	85.900253	135.112557
Linear Regression	Normal	General	285	0.179889	0.283002	84.901118	133.303556
Linear Regression	Normal	General	300	0.186699	0.292527	86.022736	134.633447
Linear Regression	Normal	General	315	0.194178	0.29834	87.387098	134.035402
Linear Regression	Normal	General	330	0.198814	0.299806	88.022462	132.530033
Linear Regression	Normal	General	345	0.202826	0.302152	87.314259	130.021874
Linear Regression	Normal	General	Global	0.154089	0.247595	81.454246	131.399079

Tabla 24: Modelo general con la aproximación general por para Regresión Lineal

Datos del modelo por vecindad con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Linear Regression	Vecindad	General	15	0.107087	0.188116	68.699127	120.667407
Linear Regression	Vecindad	General	30	0.113057	0.195321	71.843863	124.097428
Linear Regression	Vecindad	General	45	0.118489	0.202226	74.309833	126.786391
Linear Regression	Vecindad	General	60	0.122904	0.20868	75.875702	128.790475
Linear Regression	Vecindad	General	75	0.127627	0.216032	77.430117	131.053717
Linear Regression	Vecindad	General	90	0.13333	0.224561	79.302221	133.56123
Linear Regression	Vecindad	General	105	0.134907	0.224172	78.808346	130.955763
Linear Regression	Vecindad	General	120	0.136777	0.226862	78.493282	130.175777
Linear Regression	Vecindad	General	135	0.141232	0.231761	79.804608	130.927906
Linear Regression	Vecindad	General	150	0.141392	0.228957	78.66843	127.370819
Linear Regression	Vecindad	General	165	0.145317	0.234429	79.958669	128.958273
Linear Regression	Vecindad	General	180	0.148183	0.237512	80.554929	129.042157
Linear Regression	Vecindad	General	195	0.152683	0.245512	81.846917	131.512777
Linear Regression	Vecindad	General	210	0.156022	0.24971	82.160268	131.38269
Linear Regression	Vecindad	General	225	0.160632	0.256478	82.975875	132.350796
Linear Regression	Vecindad	General	240	0.167211	0.264385	84.606609	133.617775
Linear Regression	Vecindad	General	255	0.173632	0.272144	85.344442	133.546204
Linear Regression	Vecindad	General	270	0.177426	0.278959	85.554102	134.27148
Linear Regression	Vecindad	General	285	0.179919	0.283231	84.903534	133.39745
Linear Regression	Vecindad	General	300	0.185633	0.289285	85.496411	133.06593
Linear Regression	Vecindad	General	315	0.193781	0.297044	87.207218	133.445528
Linear Regression	Vecindad	General	330	0.199278	0.299541	88.243665	132.406338
Linear Regression	Vecindad	General	345	0.203029	0.301073	87.387633	129.481383
Linear Regression	Vecindad	General	Global	0.153024	0.245913	80.846774	130.472421

Tabla 25: Modelo por vecindad con la aproximación general para Regresión Lineal

SVM con kernel lineal

Datos del modelo general con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Linear Kernel	Normal	Horizons	15	0.0977798	0.2054196	62.6667475	131.6571142
SVM Linear Kernel	Normal	Horizons	30	0.1069116	0.2080881	67.8723366	132.1115787
SVM Linear Kernel	Normal	Horizons	45	0.1143579	0.2141562	71.6644661	134.1740545
SVM Linear Kernel	Normal	Horizons	60	0.1180756	0.2216354	72.8334503	136.6930230
SVM Linear Kernel	Normal	Horizons	75	0.1247359	0.2275252	75.6319431	137.9957961
SVM Linear Kernel	Normal	Horizons	90	0.1299484	0.2361777	77.2639162	140.4501795
SVM Linear Kernel	Normal	Horizons	105	0.1307939	0.2353864	76.3755179	137.4781997
SVM Linear Kernel	Normal	Horizons	120	0.1332776	0.2358043	76.4592305	135.3054469
SVM Linear Kernel	Normal	Horizons	135	0.1394702	0.2388255	78.7759623	134.8862534
SVM Linear Kernel	Normal	Horizons	150	0.1379750	0.2352852	76.7398638	130.8870612
SVM Linear Kernel	Normal	Horizons	165	0.1392535	0.2389586	76.5913446	131.4433030
SVM Linear Kernel	Normal	Horizons	180	0.1457692	0.2452593	79.2140920	133.2200267
SVM Linear Kernel	Normal	Horizons	195	0.1502447	0.2524652	80.5004566	135.2076700
SVM Linear Kernel	Normal	Horizons	210	0.1511033	0.2565119	79.5068643	134.8922708
SVM Linear Kernel	Normal	Horizons	225	0.1570414	0.2659961	81.0594287	137.2104552
SVM Linear Kernel	Normal	Horizons	240	0.1636066	0.2726046	82.6825914	137.6338730
SVM Linear Kernel	Normal	Horizons	255	0.1718494	0.2829660	84.4104549	138.7650868
SVM Linear Kernel	Normal	Horizons	270	0.1723916	0.2914599	83.0348358	140.1766185
SVM Linear Kernel	Normal	Horizons	285	0.1725865	0.2889268	81.3435590	135.9689165
SVM Linear Kernel	Normal	Horizons	300	0.1804090	0.2993502	83.0857770	137.7800485
SVM Linear Kernel	Normal	Horizons	315	0.1877855	0.3085184	84.4380579	138.5898802
SVM Linear Kernel	Normal	Horizons	330	0.1995145	0.3198218	88.1478621	141.2259372
SVM Linear Kernel	Normal	Horizons	345	0.2104340	0.3348472	90.3414418	143.8095766
SVM Linear Kernel	Normal	Horizons	Global	0.1493615	0.2572169	78.7234870	136.4157552

Tabla 26: Modelo general con la aproximación por horizontes para SVM Lineal

Datos del modelo por vecindad con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Linear Kernel	Vecindad	Horizons	15	0.0983627	0.2036828	63.0446534	130.5460262
SVM Linear Kernel	Vecindad	Horizons	30	0.1068912	0.2065409	67.8757475	131.1330075
SVM Linear Kernel	Vecindad	Horizons	45	0.1151364	0.2126120	72.1631659	133.2051106
SVM Linear Kernel	Vecindad	Horizons	60	0.1184440	0.2186452	73.0634831	134.8446545
SVM Linear Kernel	Vecindad	Horizons	75	0.1244221	0.2247727	75.4455846	136.3111909
SVM Linear Kernel	Vecindad	Horizons	90	0.1294947	0.2335825	77.0009727	138.9180151
SVM Linear Kernel	Vecindad	Horizons	105	0.1310134	0.2333015	76.5063598	136.2692198
SVM Linear Kernel	Vecindad	Horizons	120	0.1331129	0.2333966	76.3659996	133.9092912
SVM Linear Kernel	Vecindad	Horizons	135	0.1384366	0.2365020	78.1912830	133.5857289
SVM Linear Kernel	Vecindad	Horizons	150	0.1385371	0.2341905	77.0472039	130.2523050
SVM Linear Kernel	Vecindad	Horizons	165	0.1396432	0.2387192	76.8041323	131.3301996
SVM Linear Kernel	Vecindad	Horizons	180	0.1452566	0.2430367	78.9587412	132.0550317
SVM Linear Kernel	Vecindad	Horizons	195	0.1497085	0.2501358	80.2214130	133.9476554
SVM Linear Kernel	Vecindad	Horizons	210	0.1510463	0.2552053	79.4947604	134.2596216
SVM Linear Kernel	Vecindad	Horizons	225	0.1574764	0.2645218	81.2835691	136.4359305
SVM Linear Kernel	Vecindad	Horizons	240	0.1634557	0.2720331	82.5713428	137.3032747
SVM Linear Kernel	Vecindad	Horizons	255	0.1727705	0.2818724	84.8403244	138.1726757
SVM Linear Kernel	Vecindad	Horizons	270	0.1724425	0.2902361	83.0735726	139.5839514
SVM Linear Kernel	Vecindad	Horizons	285	0.1724426	0.2868660	81.3994795	135.1877051
SVM Linear Kernel	Vecindad	Horizons	300	0.1816175	0.2983515	83.6509828	137.3422479
SVM Linear Kernel	Vecindad	Horizons	315	0.1877712	0.3050140	84.4876747	137.1221187
SVM Linear Kernel	Vecindad	Horizons	330	0.2005572	0.3186361	88.6251801	140.7590323
SVM Linear Kernel	Vecindad	Horizons	345	0.2108290	0.3321207	90.4707981	142.5307198
SVM Linear Kernel	Vecindad	Horizons	Global	0.1495160	0.2553902	78.8081054	135.4349876

Tabla 27: Modelo por vecindad con la aproximación por horizontes para SVM Lineal

Datos del modelo general con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Linear Kernel	Normal	General	15	0.09943	0.193544	63.749992	124.09565
SVM Linear Kernel	Normal	General	30	0.107389	0.203	68.194854	128.913428
SVM Linear Kernel	Normal	General	45	0.11432	0.212115	71.653918	132.939608
SVM Linear Kernel	Normal	General	60	0.118406	0.21865	73.047846	134.876779
SVM Linear Kernel	Normal	General	75	0.124588	0.22721	75.546852	137.814583
SVM Linear Kernel	Normal	General	90	0.129718	0.236108	77.122975	140.39055
SVM Linear Kernel	Normal	General	105	0.13078	0.23468	76.370099	137.058154
SVM Linear Kernel	Normal	General	120	0.133273	0.238427	76.463894	136.808441
SVM Linear Kernel	Normal	General	135	0.139482	0.245123	78.788079	138.453529
SVM Linear Kernel	Normal	General	150	0.137757	0.239644	76.627892	133.330247
SVM Linear Kernel	Normal	General	165	0.139427	0.241701	76.687854	132.968269
SVM Linear Kernel	Normal	General	180	0.145475	0.24914	79.056716	135.351437
SVM Linear Kernel	Normal	General	195	0.150165	0.255991	80.444581	137.083797
SVM Linear Kernel	Normal	General	210	0.151447	0.259139	79.672291	136.277155
SVM Linear Kernel	Normal	General	225	0.156744	0.268243	80.918297	138.417617
SVM Linear Kernel	Normal	General	240	0.163491	0.27397	82.635102	138.404708
SVM Linear Kernel	Normal	General	255	0.170342	0.284476	83.670321	139.618783
SVM Linear Kernel	Normal	General	270	0.172046	0.2912	82.89799	140.167334
SVM Linear Kernel	Normal	General	285	0.173952	0.291169	82.091938	137.270812
SVM Linear Kernel	Normal	General	300	0.17965	0.299378	82.798544	137.937058
SVM Linear Kernel	Normal	General	315	0.187222	0.305283	84.304112	137.386793
SVM Linear Kernel	Normal	General	330	0.19343	0.309798	85.70148	137.160598
SVM Linear Kernel	Normal	General	345	0.197035	0.310117	84.915801	133.754753
SVM Linear Kernel	Normal	General	Global	0.148503	0.256005	78.407019	135.933917

Tabla 28: Modelo general con la aproximación general para SVM Lineal

Datos del modelo por vecindad con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Linear Kernel	Vecindad	General	15	0.099931	0.193768	64.06389	124.232983
SVM Linear Kernel	Vecindad	General	30	0.107413	0.20207	68.210461	128.336927
SVM Linear Kernel	Vecindad	General	45	0.114099	0.21044	71.506577	131.885469
SVM Linear Kernel	Vecindad	General	60	0.118015	0.216231	72.801491	133.389874
SVM Linear Kernel	Vecindad	General	75	0.124185	0.225145	75.289614	136.548459
SVM Linear Kernel	Vecindad	General	90	0.1295	0.233986	76.997514	139.145347
SVM Linear Kernel	Vecindad	General	105	0.13074	0.23293	76.344978	136.03722
SVM Linear Kernel	Vecindad	General	120	0.133334	0.236196	76.491399	135.515735
SVM Linear Kernel	Vecindad	General	135	0.138913	0.242556	78.461363	137.004215
SVM Linear Kernel	Vecindad	General	150	0.137917	0.237611	76.709136	132.178624
SVM Linear Kernel	Vecindad	General	165	0.140476	0.241245	77.260483	132.709765
SVM Linear Kernel	Vecindad	General	180	0.145261	0.246165	78.935522	133.728781
SVM Linear Kernel	Vecindad	General	195	0.150028	0.253911	80.372984	135.966546
SVM Linear Kernel	Vecindad	General	210	0.151468	0.257055	79.69168	135.192278
SVM Linear Kernel	Vecindad	General	225	0.156464	0.265296	80.760474	136.854964
SVM Linear Kernel	Vecindad	General	240	0.163237	0.272985	82.510105	137.902406
SVM Linear Kernel	Vecindad	General	255	0.170312	0.281626	83.63635	138.171898
SVM Linear Kernel	Vecindad	General	270	0.172321	0.289211	83.035207	139.194533
SVM Linear Kernel	Vecindad	General	285	0.174814	0.29057	82.48456	136.949593
SVM Linear Kernel	Vecindad	General	300	0.179785	0.296008	82.821991	136.304009
SVM Linear Kernel	Vecindad	General	315	0.187727	0.303338	84.534094	136.481333
SVM Linear Kernel	Vecindad	General	330	0.194412	0.308926	86.12674	136.728221
SVM Linear Kernel	Vecindad	General	345	0.197536	0.308093	85.116695	132.808471
SVM Linear Kernel	Vecindad	General	Global	0.148604	0.254146	78.441883	134.92468

Tabla 29: Modelo por vecindad con la aproximación general para SVM Lineal

SVM con kernel Radial

Datos del modelo general con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Radial Kernel	Normal	Horizons	15	0.1214757	0.1911313	77.9573479	122.5930244
SVM Radial Kernel	Normal	Horizons	30	0.1241511	0.1989141	78.9232509	126.3543822
SVM Radial Kernel	Normal	Horizons	45	0.1300502	0.2068591	81.5966932	129.6901141
SVM Radial Kernel	Normal	Horizons	60	0.1323807	0.2139181	81.7391209	132.0001236
SVM Radial Kernel	Normal	Horizons	75	0.1391451	0.2206723	84.4562850	133.8948371
SVM Radial Kernel	Normal	Horizons	90	0.1464684	0.2289837	87.1466662	136.1823320
SVM Radial Kernel	Normal	Horizons	105	0.1430374	0.2275248	83.5654066	132.9055652
SVM Radial Kernel	Normal	Horizons	120	0.1494730	0.2310523	85.8074093	132.5945695
SVM Radial Kernel	Normal	Horizons	135	0.1546153	0.2355722	87.3903901	133.0762306
SVM Radial Kernel	Normal	Horizons	150	0.1533737	0.2319480	85.3571607	129.0396561
SVM Radial Kernel	Normal	Horizons	165	0.1581953	0.2356039	87.1067604	129.6241430
SVM Radial Kernel	Normal	Horizons	180	0.1665729	0.2427782	90.5807031	131.9133490
SVM Radial Kernel	Normal	Horizons	195	0.1698168	0.2488916	91.0193318	133.3193921
SVM Radial Kernel	Normal	Horizons	210	0.1751969	0.2552278	92.2347869	134.2768769
SVM Radial Kernel	Normal	Horizons	225	0.1824941	0.2624252	94.2693618	135.4398871
SVM Radial Kernel	Normal	Horizons	240	0.1873213	0.2676624	94.8224174	135.2239225
SVM Radial Kernel	Normal	Horizons	255	0.1948680	0.2784123	95.8128434	136.6149153
SVM Radial Kernel	Normal	Horizons	270	0.1998637	0.2846811	96.3570966	136.9503085
SVM Radial Kernel	Normal	Horizons	285	0.1925802	0.2856296	90.8355990	134.4572117
SVM Radial Kernel	Normal	Horizons	300	0.2112670	0.2980322	97.2952539	137.0094565
SVM Radial Kernel	Normal	Horizons	315	0.2219578	0.3071660	99.7334076	137.7256424
SVM Radial Kernel	Normal	Horizons	330	0.2282097	0.3144047	100.8212554	138.5473310
SVM Radial Kernel	Normal	Horizons	345	0.2330413	0.3199329	100.0048945	137.0271896
SVM Radial Kernel	Normal	Horizons	Global	0.1702415	0.2516271	89.7753671	133.3243678

Tabla 30: Modelo general con la aproximación por horizontes para SVM Radial

Datos del modelo por vecindad con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Radial Kernel	Vecindad	Horizons	15	0.1159449	0.1885142	74.4270737	120.9202736
SVM Radial Kernel	Vecindad	Horizons	30	0.1183648	0.1953215	75.2415926	124.0870694
SVM Radial Kernel	Vecindad	Horizons	45	0.1234730	0.2027941	77.4685585	127.1394686
SVM Radial Kernel	Vecindad	Horizons	60	0.1261271	0.2097534	77.8795845	129.4407872
SVM Radial Kernel	Vecindad	Horizons	75	0.1326213	0.2175453	80.4866662	131.9773920
SVM Radial Kernel	Vecindad	Horizons	90	0.1410470	0.2269641	83.9146350	134.9937550
SVM Radial Kernel	Vecindad	Horizons	105	0.1404013	0.2260409	82.0255548	132.0425870
SVM Radial Kernel	Vecindad	Horizons	120	0.1433886	0.2282742	82.3167450	130.9888041
SVM Radial Kernel	Vecindad	Horizons	135	0.1498889	0.2330472	84.7271706	131.6477263
SVM Radial Kernel	Vecindad	Horizons	150	0.1483651	0.2298848	82.5836694	127.8865958
SVM Radial Kernel	Vecindad	Horizons	165	0.1542042	0.2349681	84.8903706	129.2650796
SVM Radial Kernel	Vecindad	Horizons	180	0.1587255	0.2388317	86.3356103	129.7865350
SVM Radial Kernel	Vecindad	Horizons	195	0.1647505	0.2465764	88.2946359	132.0946040
SVM Radial Kernel	Vecindad	Horizons	210	0.1695868	0.2510210	89.3289023	132.1046280
SVM Radial Kernel	Vecindad	Horizons	225	0.1751205	0.2591348	90.4406759	133.7310636
SVM Radial Kernel	Vecindad	Horizons	240	0.1819991	0.2657215	92.1283874	134.2598149
SVM Radial Kernel	Vecindad	Horizons	255	0.1879068	0.2754644	92.3910649	135.1572163
SVM Radial Kernel	Vecindad	Horizons	270	0.1940057	0.2832684	93.5213954	136.2819745
SVM Radial Kernel	Vecindad	Horizons	285	0.1904646	0.2834958	89.9037029	133.4846124
SVM Radial Kernel	Vecindad	Horizons	300	0.2035586	0.2934538	93.7869564	134.9805953
SVM Radial Kernel	Vecindad	Horizons	315	0.2141831	0.3025930	96.3366831	135.8371393
SVM Radial Kernel	Vecindad	Horizons	330	0.2214996	0.3137350	97.8796809	138.3500117
SVM Radial Kernel	Vecindad	Horizons	345	0.2276921	0.3182945	97.7768632	136.4827031
SVM Radial Kernel	Vecindad	Horizons	Global	0.1644921	0.2488999	86.6993991	131.8669755

Tabla 31: Modelo por vecindad con la aproximación por horizontes para SVM Radial

Datos del modelo general con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Radial Kernel	Normal	General	15	0.118084	0.190673	75.78612074	122.323386
SVM Radial Kernel	Normal	General	30	0.123864	0.199025	78.73379281	126.450272
SVM Radial Kernel	Normal	General	45	0.129396	0.206156	81.18038122	129.263164
SVM Radial Kernel	Normal	General	60	0.133592	0.213066	82.49726028	131.496682
SVM Radial Kernel	Normal	General	75	0.138676	0.220518	84.17263132	133.80585
SVM Radial Kernel	Normal	General	90	0.14356	0.228152	85.40540242	135.690724
SVM Radial Kernel	Normal	General	105	0.145472	0.228019	85.00583213	133.209334
SVM Radial Kernel	Normal	General	120	0.147665	0.230956	84.78117645	132.551646
SVM Radial Kernel	Normal	General	135	0.152818	0.23607	86.38485592	133.372035
SVM Radial Kernel	Normal	General	150	0.15228	0.232486	84.76455136	129.361845
SVM Radial Kernel	Normal	General	165	0.15491	0.235691	85.28373098	129.681397
SVM Radial Kernel	Normal	General	180	0.159271	0.241852	86.62670313	131.416676
SVM Radial Kernel	Normal	General	195	0.163176	0.248155	87.49873324	132.931808
SVM Radial Kernel	Normal	General	210	0.165969	0.252632	87.42316179	132.920755
SVM Radial Kernel	Normal	General	225	0.170628	0.260128	88.1974285	134.284185
SVM Radial Kernel	Normal	General	240	0.177154	0.265934	89.67039988	134.392631
SVM Radial Kernel	Normal	General	255	0.183203	0.275008	90.12895365	135.018375
SVM Radial Kernel	Normal	General	270	0.18622	0.281067	89.85388414	135.311866
SVM Radial Kernel	Normal	General	285	0.187829	0.283626	88.71037874	133.639509
SVM Radial Kernel	Normal	General	300	0.194567	0.293047	89.72813968	134.932694
SVM Radial Kernel	Normal	General	315	0.20199	0.29855	90.984159	134.198086
SVM Radial Kernel	Normal	General	330	0.206703	0.300515	91.59891732	132.927215
SVM Radial Kernel	Normal	General	345	0.210313	0.302638	90.62321526	130.306557
SVM Radial Kernel	Normal	General	Global	0.162928	0.248868	86.30607869	132.151595

Tabla 32: Modelo general con la aproximación general para SVM Radial

Datos del modelo por vecindad con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
SVM Radial Kernel	Vecindad	General	15	0.112384	0.188961	72.119116	121.218765
SVM Radial Kernel	Vecindad	General	30	0.118774	0.196244	75.497902	124.693318
SVM Radial Kernel	Vecindad	General	45	0.124271	0.203179	77.957919	127.393489
SVM Radial Kernel	Vecindad	General	60	0.12849	0.20969	79.345776	129.423899
SVM Radial Kernel	Vecindad	General	75	0.133568	0.217276	81.05933	131.822833
SVM Radial Kernel	Vecindad	General	90	0.139203	0.225755	82.81561	134.282511
SVM Radial Kernel	Vecindad	General	105	0.140962	0.225569	82.367647	131.782257
SVM Radial Kernel	Vecindad	General	120	0.143082	0.228496	82.137406	131.126622
SVM Radial Kernel	Vecindad	General	135	0.147669	0.233353	83.465513	131.83647
SVM Radial Kernel	Vecindad	General	150	0.147845	0.23035	82.282778	128.156774
SVM Radial Kernel	Vecindad	General	165	0.151705	0.235588	83.504035	129.614692
SVM Radial Kernel	Vecindad	General	180	0.154872	0.238977	84.226641	129.856231
SVM Radial Kernel	Vecindad	General	195	0.159125	0.246565	85.326478	132.089193
SVM Radial Kernel	Vecindad	General	210	0.162218	0.250653	85.453915	131.902115
SVM Radial Kernel	Vecindad	General	225	0.166809	0.257613	86.194009	132.955199
SVM Radial Kernel	Vecindad	General	240	0.173533	0.26535	87.844266	134.125309
SVM Radial Kernel	Vecindad	General	255	0.17952	0.272589	88.283696	133.806086
SVM Radial Kernel	Vecindad	General	270	0.183134	0.279667	88.354048	134.655891
SVM Radial Kernel	Vecindad	General	285	0.185468	0.283816	87.575096	133.718967
SVM Radial Kernel	Vecindad	General	300	0.19126	0.289962	88.161168	133.447913
SVM Radial Kernel	Vecindad	General	315	0.199347	0.297365	89.789792	133.673464
SVM Radial Kernel	Vecindad	General	330	0.204778	0.300448	90.76021	132.89747
SVM Radial Kernel	Vecindad	General	345	0.208551	0.302062	89.857556	130.007527
SVM Radial Kernel	Vecindad	General	Global	0.158981	0.246936	84.103474	131.064652

Tabla 33: Modelo por vecindad con la aproximación general para SVM Radial

Random Forest

Datos del modelo general con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Random Forest	Normal	Horizons	15	0.0988953	0.1896394	63.4366035	121.6215307
Random Forest	Normal	Horizons	30	0.1088941	0.1996440	69.1677709	126.7772656
Random Forest	Normal	Horizons	45	0.1172661	0.2103355	73.5528578	131.8641685
Random Forest	Normal	Horizons	60	0.1217006	0.2157552	75.0978923	133.1300919
Random Forest	Normal	Horizons	75	0.1295186	0.2276797	78.5683395	138.0847841
Random Forest	Normal	Horizons	90	0.1330622	0.2311925	79.1431235	137.5038833
Random Forest	Normal	Horizons	105	0.1365845	0.2340810	79.7623374	136.7121447
Random Forest	Normal	Horizons	120	0.1383533	0.2360438	79.4179253	135.4861559
Random Forest	Normal	Horizons	135	0.1430393	0.2398921	80.8181621	135.5210784
Random Forest	Normal	Horizons	150	0.1418641	0.2381878	78.8986879	132.4717284
Random Forest	Normal	Horizons	165	0.1447701	0.2411090	79.6242335	132.5963117
Random Forest	Normal	Horizons	180	0.1506771	0.2472813	81.8545870	134.3211865
Random Forest	Normal	Horizons	195	0.1579297	0.2607204	84.5881409	139.6326827
Random Forest	Normal	Horizons	210	0.1581541	0.2588922	83.2397043	136.2842973
Random Forest	Normal	Horizons	225	0.1683411	0.2729984	86.9053257	140.9206573
Random Forest	Normal	Horizons	240	0.1716476	0.2775672	86.7720822	140.2481668
Random Forest	Normal	Horizons	255	0.1792885	0.2880977	88.0345108	141.4448350
Random Forest	Normal	Horizons	270	0.1817880	0.2949603	87.5586679	142.0020076
Random Forest	Normal	Horizons	285	0.1781259	0.2865119	83.8786624	134.8202984
Random Forest	Normal	Horizons	300	0.1934685	0.3097710	89.0630277	142.5708174
Random Forest	Normal	Horizons	315	0.2047119	0.3187423	91.9311433	142.9522911
Random Forest	Normal	Horizons	330	0.2097023	0.3244751	92.7431493	143.4080244
Random Forest	Normal	Horizons	345	0.2181513	0.3380719	93.7468962	145.2481090
Random Forest	Normal	Horizons	Global	0.1559102	0.2583326	82.0784275	136.7661964

Tabla 34: Modelo general con la aproximación por horizontes para Random Forest

Datos del modelo por vecindad con la aproximación por horizontes

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Random Forest	Vecindad	Horizons	15	0.0968427	0.1860020	62.1170918	119.2979952
Random Forest	Vecindad	Horizons	30	0.1066921	0.1960709	67.7751189	124.5458659
Random Forest	Vecindad	Horizons	45	0.1150359	0.2057845	72.1432403	129.0028059
Random Forest	Vecindad	Horizons	60	0.1200944	0.2142029	74.1022438	132.1730736
Random Forest	Vecindad	Horizons	75	0.1260041	0.2204695	76.4350881	133.7387640
Random Forest	Vecindad	Horizons	90	0.1327085	0.2292525	78.9253215	136.3364856
Random Forest	Vecindad	Horizons	105	0.1344936	0.2295440	78.5324337	134.0585471
Random Forest	Vecindad	Horizons	120	0.1361430	0.2307068	78.1206178	132.3905375
Random Forest	Vecindad	Horizons	135	0.1408043	0.2350368	79.5604996	132.7992608
Random Forest	Vecindad	Horizons	150	0.1398450	0.2339295	77.7801091	130.1009966
Random Forest	Vecindad	Horizons	165	0.1420810	0.2369882	78.1443479	130.3706938
Random Forest	Vecindad	Horizons	180	0.1477420	0.2417507	80.2758086	131.3560586
Random Forest	Vecindad	Horizons	195	0.1553874	0.2539755	83.2370773	136.0579051
Random Forest	Vecindad	Horizons	210	0.1584261	0.2597169	83.4061230	136.6752976
Random Forest	Vecindad	Horizons	225	0.1638171	0.2665789	84.5595220	137.5977167
Random Forest	Vecindad	Horizons	240	0.1670832	0.2690698	84.4601745	135.9435480
Random Forest	Vecindad	Horizons	255	0.1748661	0.2793344	85.8595564	137.0803716
Random Forest	Vecindad	Horizons	270	0.1782427	0.2846993	85.8433307	136.9768362
Random Forest	Vecindad	Horizons	285	0.1770814	0.2833906	83.4598397	133.5181062
Random Forest	Vecindad	Horizons	300	0.1863803	0.2968729	85.8265492	136.6281296
Random Forest	Vecindad	Horizons	315	0.2001772	0.3086735	89.9817787	138.6699172
Random Forest	Vecindad	Horizons	330	0.2032558	0.3141253	89.9208417	138.8343708
Random Forest	Vecindad	Horizons	345	0.2112646	0.3223402	90.9315000	138.5946934
Random Forest	Vecindad	Horizons	Global	0.1528030	0.2521094	80.4955745	133.5977381

Tabla 35: Modelo por vecindad con la aproximación por horizontes para Random Forest

Datos del modelo general con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Random Forest	Normal	General	15	0.108336	0.196361	69.51269135	125.990122
Random Forest	Normal	General	30	0.115098	0.20535	73.1198632	130.438373
Random Forest	Normal	General	45	0.119871	0.210567	75.19678309	132.059088
Random Forest	Normal	General	60	0.126616	0.220281	78.14568925	135.960672
Random Forest	Normal	General	75	0.132223	0.228453	80.21531898	138.617873
Random Forest	Normal	General	90	0.135462	0.234284	80.5561792	139.339761
Random Forest	Normal	General	105	0.138445	0.23657	80.89002537	138.247195
Random Forest	Normal	General	120	0.141855	0.241394	81.39813053	138.519765
Random Forest	Normal	General	135	0.145463	0.245545	82.18607974	138.72251
Random Forest	Normal	General	150	0.144618	0.239443	80.4649065	133.256958
Random Forest	Normal	General	165	0.148822	0.246794	81.87266031	135.816445
Random Forest	Normal	General	180	0.153329	0.251799	83.33452948	136.838847
Random Forest	Normal	General	195	0.156659	0.256103	83.91251856	137.171216
Random Forest	Normal	General	210	0.159143	0.260763	83.70925255	137.133672
Random Forest	Normal	General	225	0.163955	0.265503	84.62110746	136.942199
Random Forest	Normal	General	240	0.172283	0.275938	87.09999038	139.351801
Random Forest	Normal	General	255	0.177577	0.284626	87.19371137	139.676212
Random Forest	Normal	General	270	0.179513	0.288782	86.44868149	138.935581
Random Forest	Normal	General	285	0.185495	0.294303	87.29314937	138.349045
Random Forest	Normal	General	300	0.19187	0.306436	88.29794434	140.940894
Random Forest	Normal	General	315	0.197717	0.308199	88.90664077	138.442134
Random Forest	Normal	General	330	0.20596	0.312853	91.11002167	138.207195
Random Forest	Normal	General	345	0.206545	0.317122	88.75269391	136.132816
Random Forest	Normal	General	Global	0.15682	0.257716	82.79298126	136.74306

Tabla 36: Modelo general con la aproximación general para Random Forest

Datos del modelo por vecindad con la aproximación general

Algorithm	Architecture	Approach	Horizon	nMAE	nRMSE	MAE	RMSE
Random Forest	Vecindad	General	15	0.104414	0.1896	66.994237	121.658007
Random Forest	Vecindad	General	30	0.111345	0.199055	70.746601	126.469113
Random Forest	Vecindad	General	45	0.117327	0.206698	73.592847	129.628908
Random Forest	Vecindad	General	60	0.122502	0.214187	75.626966	132.243535
Random Forest	Vecindad	General	75	0.126901	0.219116	76.96713	132.89656
Random Forest	Vecindad	General	90	0.131756	0.227385	78.359647	135.255618
Random Forest	Vecindad	General	105	0.133739	0.227863	78.149293	133.185676
Random Forest	Vecindad	General	120	0.135984	0.232299	78.060138	133.351326
Random Forest	Vecindad	General	135	0.140049	0.237794	79.124996	134.339829
Random Forest	Vecindad	General	150	0.139968	0.23358	77.868549	129.958102
Random Forest	Vecindad	General	165	0.142929	0.236972	78.613169	130.372427
Random Forest	Vecindad	General	180	0.146322	0.24216	79.507004	131.570067
Random Forest	Vecindad	General	195	0.151099	0.249573	80.974185	133.702975
Random Forest	Vecindad	General	210	0.154661	0.254596	81.40461	133.936528
Random Forest	Vecindad	General	225	0.158457	0.259829	81.813022	134.068092
Random Forest	Vecindad	General	240	0.165486	0.268531	83.684615	135.655208
Random Forest	Vecindad	General	255	0.169743	0.273348	83.405564	134.196749
Random Forest	Vecindad	General	270	0.172652	0.278601	83.139548	134.039491
Random Forest	Vecindad	General	285	0.177471	0.285477	83.592827	134.290586
Random Forest	Vecindad	General	300	0.18638	0.298364	85.772469	137.206991
Random Forest	Vecindad	General	315	0.196664	0.306082	88.437313	137.582729
Random Forest	Vecindad	General	330	0.197474	0.302165	87.333857	133.522281
Random Forest	Vecindad	General	345	0.200413	0.305283	86.224715	131.340821
Random Forest	Vecindad	General	Global	0.151467	0.249937	79.973622	132.629201

Tabla 37: Modelo por vecindad con la aproximación general para Random Forest

Anexo C- Summary

Introduction

In the last decades, technology has experienced a great growth. Thanks to the internet, we have more processing capability and access to educational resources. Although the field of artificial intelligence is not new, it is a good time to look for different approaches to current problems.

In this case our problem is the prediction of solar irradiance. This is related to the power production capacity of solar power plants. If the accuracy predicting irradiance could be increased, better production estimates could be done. Currently this is done using physical models, machine learning techniques will be applied and tested, such as cross-validation or hyperparameter optimisation among others, on these models to see if we can obtain better results in the prediction.

Solar irradiance is the set of electromagnetic waves emitted by the sun. This irradiance can be measured in space or on earth's surface. If it is measured on earth's surface, there are factors that affect its value, such as what position the sun is in at that time, or weather conditions. There are several types of measurements on solar irradiance, but this study will focus on global horizontal irradiance.

It is easy to see how important this concept is in the production of electrical energy. That's where photovoltaic solar energy comes in. This energy comes from the transformation of solar irradiance into electricity using photovoltaic technology.

Having these concepts in mind, the relevance of this field of study can be seen. If it could predict solar irradiance, it could determine the amount of energy that is produced in a photovoltaic solar power plant.

Study approach

The proposal for the new model that is going to be presented will be developed throughout this document. As previously indicated, this is a new approach. Now we are going to present the two models that we are going to use for this study.

General Model

The general model consists in using the values that the physical models have predicted for a horizon as inputs to make predictions. This is represented in the following equation (Eq. 13):

$$f(t,h) = f(P_1(t,h), P_2(t,h), P_3(t,h), P_4(t,h)) \quad (13)$$

Ecuación 13: General model

Neighbourhood model

The neighbourhood model consists in using the values of the temporally adjacent horizons of each of the physical models to complement the data in that predictor, that is, the general formula of the prediction would be (Eq. 14):

$$f(t,h) = f\begin{pmatrix} P_1(t,h-15), P_1(t,h), P_1(t,h+15), \dots, \\ P_4(t,h-15), P_4(t,h), P_4(t,h+15) \end{pmatrix} \quad (14)$$

Ecuación 14: Neighbourhood model

Machine learning techniques

At this point the explanation about the techniques used, cross validation and hyperparameters optimization of the algorithms, will be developed. It will also explain what algorithms have been used.

Cross validation

It is a widely used technique for the evaluation of models in artificial intelligence. It involves dividing a dataset into several subsets and iterating and using those same subsets as training data or as test data interchangeably. From each of these iterations, evaluation metrics will be extracted, and the result will be the average of the *n* iterations performed.

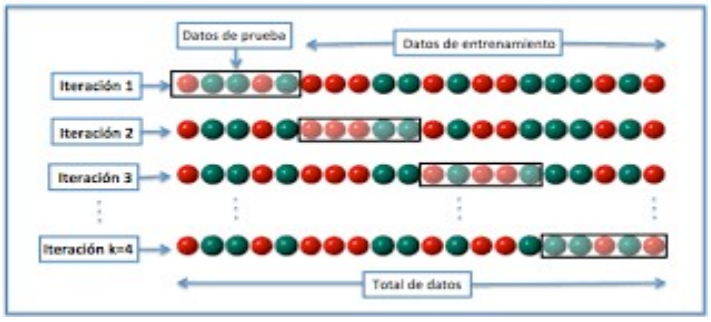

Ilustración 27: Example cross-validation with 4 iterations

As it can be seen in Figure 27, the example shows a division into 4 iterations. That means that the set has been divided into 4, and in each iteration 3 subsets make up the training data, and 1 subset the test data.

This technique is mainly used to test models that are to be developed, since it guarantees the independence of the results compared to training and test data. A problem in supervised learning can occur when the same data set is used in both training and testing. It is what is known as overfitting. This makes the algorithm have very good results, since it knows what it must predict in training, but its performance declines when it is exposed to new data to predict.

Hyperparameter optimization

This is another of the techniques that have been used for this development. One of the main problems that some of the machine learning algorithms have is that they have parameters that can be modified and affect the results of the predictions made. For example; one of the algorithms used the Random Forest has as parameters the number of trees / nodes that the forest has. These values can affect the accuracy of predictions, which is why this technique has been used. In addition, these hyperparameters are different for each problem, so each time a model is presented, it is necessary to seek for the best configuration.

The procedure is the next one, and it is applied within each of the iterations of the cross validation. In this case, for the algorithms that needed it in our development, a list has been

created with the values to be introduced in those parameters. Instead of taking the corresponding subset of training and applying it directly to the algorithm, what is done is to split that subset. In this case we create a training set and a validation set.

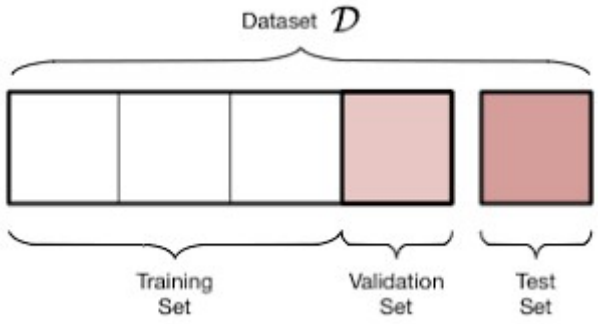

Ilustración 28: Split dataset example

As it has been explained in the previous point, we must make this new split, because if we used the previously created sets, we would have an overestimation problem. The division made in this case is random. As it only wants to find the best hyperparameters, it does matter if any record is overrepresented, but in this case, it simplifies the development and the impact that it could have over the results is not that important. Once the new split is made, the algorithm is trained with that data, and then the validation subset for the tests will be used. We will repeat this process for each of the values that have been defined in the list of parameters, and finally it will choose the one that has obtained the best results.

This process will be repeated in each of the n iterations of the cross validation since it must be done for each of the subsets. Once the best hyperparameter for that algorithm is obtained, the next step is with the complete set of training that had been defined for that iteration of the cross validation. The algorithm will be trained and with the subset of tests, the predictions will be made, applying the parameter obtained in this process to the algorithm. This ensures that overfitting is avoided, and that the algorithm is being trained with the best set of hyperparameters.

Algorithms used

At this point, after having set everything in context, it will explain what algorithms are going to be used for the tests. But first the problem must have to be define, to be able to choose these algorithms with better criteria.

To determine the type of problem ~~is~~, the data must have been analysed~~,~~ and the objectives. Observing the data, we can see that there is a time series on which we want to make predictions. It has to be established the dependency relationship that exists with our variables to make predictions. This corresponds to a regression analysis.

This is important because not all machine learning algorithms solve the same type of problems. For example, there are algorithms that better solve classification problems.

To solve this regression problem, 4 algorithms have been selected, which will be explained below:

- **Linear regression**: It is a mathematical model that allows us to approximate the dependency relationship between 2 or more variables. The equation that expresses this relationship would be the one shown below (Eq. 15):

$$f(t) = C_0 + C_1P_1 + \ldots + C_nP_n \quad (15)$$

Ecuación 15: Linear Regresión

- **Support vector machine**: (for its acronym SVM). This algorithm represents the values in a dimensional space, and tries to find the hyperplane, or hyperplanes that best split the data. The construction of these hyperplanes is determined by a main function that is also known as a kernel. Although this definition fits more closely with a classification problem, it can also be used in regression problems, as indicated. For this problem, 2 different kernels have been tested, one linear and the other with a Gaussian radial base. You can see an example in illustration 29.

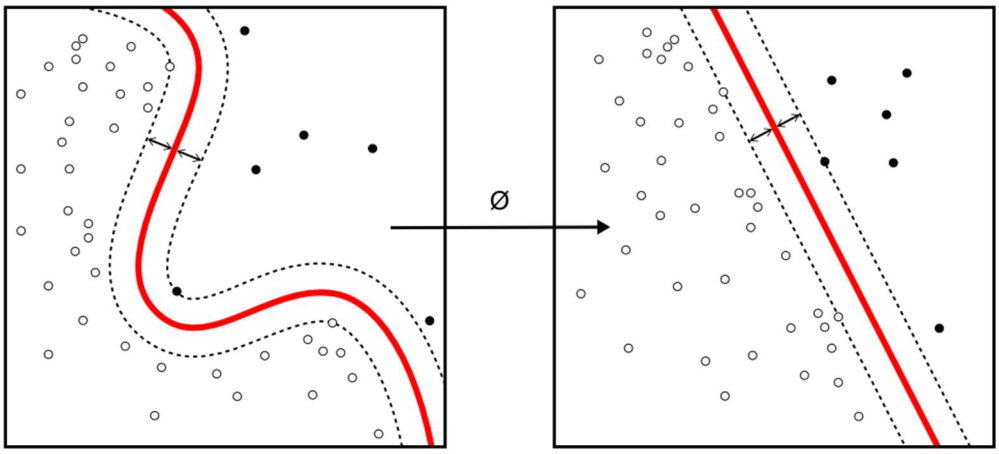

Ilustración 29: SVM example

- **Random Forest**: This algorithm builds multiple decision trees that train independently to make the prediction. A decision tree is a structure in which each node represents a decision to be made. The more nodes the more possible results can be obtained. Each of the trees that make up the forest is trained with a subset of the data chosen randomly. The result is the calculation of the average of each of the results of each decision tree.

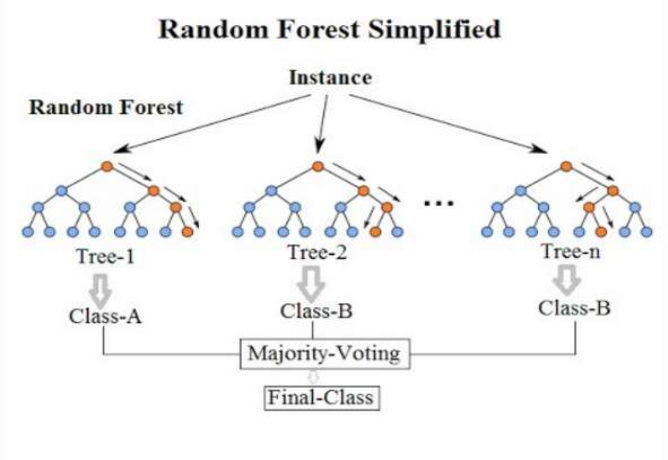

Ilustración 30: Random Forest Example

Objectives

The objective of this project is to see if the model presented is better at predicting solar irradiance than the current scenario. To do this, a general model has been used as a base scenario, which consists of introducing the values of the physical models to the machine learning algorithms and comparing it with the result of the proposed model, a neighbourhood model. For this, the following objectives must be met:

- **Programming language**: One must be used as a language that meets the following two conditions. That it is an open source language, and that it has a set of libraries, also open source, for the management of machine learning algorithms.
- **Normalization**: The data should be normalized to avoid problems when introducing them in the machine learning algorithms.
- **Cross validation**: This technique should be applied to make sure that the predictions does not have an overfit.
- **Parameter adjustment**: This technique should be applied to be able to find, in the algorithms that are necessary, the parameters that provide the best results for our assumption.
- **Algorithm training**: Once the previous steps have been completed, the algorithms will be trained, and the test sets will be used to see what results are obtained in the predictions.
- **Calculation of errors**: To check the performance in the prediction, a series of metrics will be calculated that it will use as a measure to know which hypothesis has obtained better results.
- **Storage of results**: The results obtained should be stored so that they can be interpreted and analysed in a comfortable way.

Results

The set of tests has been as follows: For each of the algorithms, 4 in total, each of the models has been tested with each of the approximations, which makes a total of 16 tests.

The results will be presented in the following way: There are 2 different models, one general, or without neighbourhood, and the neighbourhood model, there are also 2 approximations, a general approximation, which consists in having a single global prediction function, and an approximation by horizons, in which, there will be a function for each horizon to be predicted.

Global error

In this section the global error value of each of the metrics will be shown. It will be divided into 2, a section for each approach.

General approach

This is the table with the current values for this approach.

Algorithm	Architecture	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	0.1540888497	0.2475945287	81.4542455790	131.3990789257
Linear Regression	Vecindad	0.1530237866	0.2459126542	80.8467740350	130.4724214863
SVM Linear Kernel	Normal	0.1485030689	0.2560045466	78.4070186130	135.9339166109
SVM Linear Kernel	Vecindad	0.1486037612	0.2541462914	78.4418828382	134.9246803849
SVM Radial Kernel	Normal	0.1629278371	0.2488679504	86.3060786944	132.1515952640
SVM Radial Kernel	Vecindad	0.1589810816	0.2469360821	84.1034742718	131.0646519330
Random Forest	Normal	0.1568198169	0.2577160418	82.7929812553	136.7430598100
Random Forest	Vecindad	0.1514668469	0.2499372661	79.9736218161	132.6292007967

Tabla 38: Errors by algorithm in the general approach

As can be seen in the table, in the general approach, the neighbourhood model is consistently better in all cases, although the improvement is small, it has better results. In the case of the SVM algorithm with linear kernel, although it is seen that the neighbourhood model is marginally worse in the results in nMAE, it can be seen that it is better in the nRMSE, and this is because even the average of the errors be slightly smaller, the nRMSE penalizes larger errors of greater magnitude, so for this model it is more interesting to consider this error metric to a greater extent.

Horizons approach

This is the table with the current values for this approach.

Algorithm	Architecture	nMAE	nRMSE	MAE	RMSE
Linear Regression	Normal	0.15541363101	0.24932568038	81.92981942682	132.07962010413
Linear Regression	Vecindad	0.15426508400	0.24804070165	81.31179798785	131.36296184847
SVM Linear Kernel	Normal	0.14936152820	0.25721694664	78.72348697905	136.41575522645
SVM Linear Kernel	Vecindad	0.14951601472	0.25539023206	78.80810541496	135.43498757066
SVM Radial Kernel	Normal	0.17024153757	0.25162711833	89.77536706860	133.32436783367
SVM Radial Kernel	Vecindad	0.16449212257	0.24889992425	86.69939911595	131.86697549556
Random Forest	Normal	0.15591017901	0.25833259094	82.07842745842	136.76619638001
Random Forest	Vecindad	0.15280297174	0.25210936747	80.49557454683	133.59773813678

Tabla 39: Errors by algorithm in the horizons approach

In this table it can be observed that, as in the general approach, the neighbourhood model improves with respect to the general model. In case of the SVM algorithm with linear kernel it can be observed the same case as in the previous example. Although the mean of the errors is smaller, the distance between the prediction and the observation is greater, so it can be assured that the neighbourhood model presents better results.

Errors by horizon

In this section the errors by horizon will be shown. The results will be presented by algorithm, and within that division they will be presented for each of the metrics. In each graph the 2 approximations used will be presented, the general one and the one by horizons, to be able to compare them among themselves and reflect the result better.

Linear regression

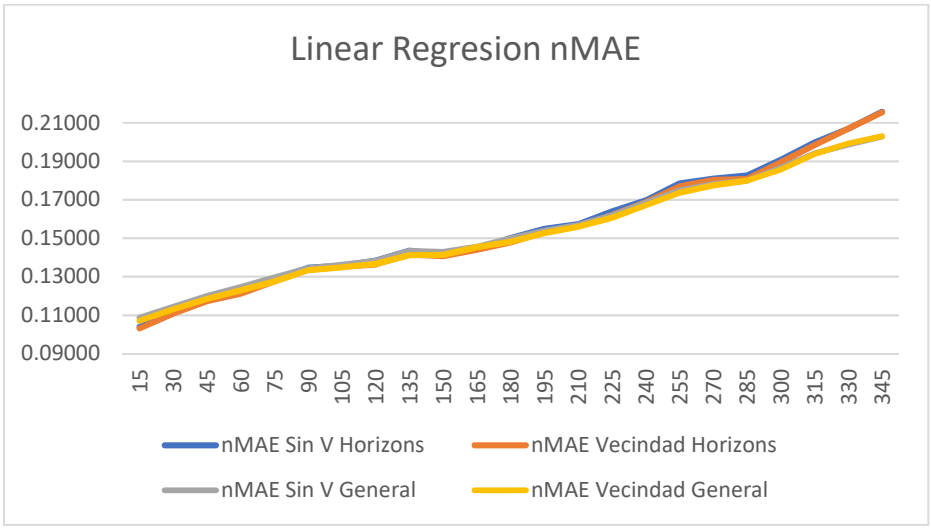

Ilustración 31: nMAE Chart for linear regression

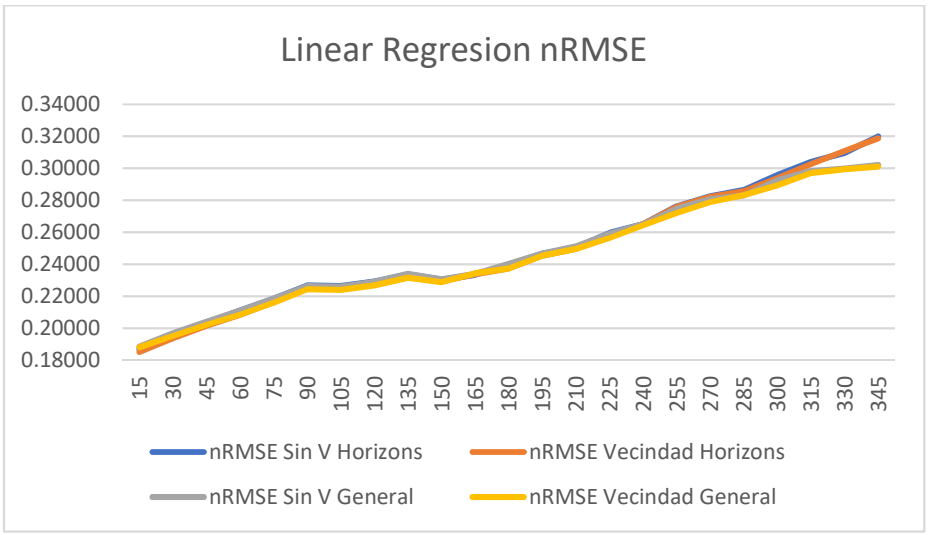

Ilustración 32: nRMSE Chart for linear regression

As we can see in illustration 31, the neighbourhood model shows better results in nearby horizons, being in both approaches, better than the normal model. It can also be seen that the architecture by horizons gets better results the closer to the time of the prediction, but in distant horizons it is not so good. However, the general architecture is somewhat better in distant horizons. This trend can also be confirmed in illustration 32.

SVM with linear kernel

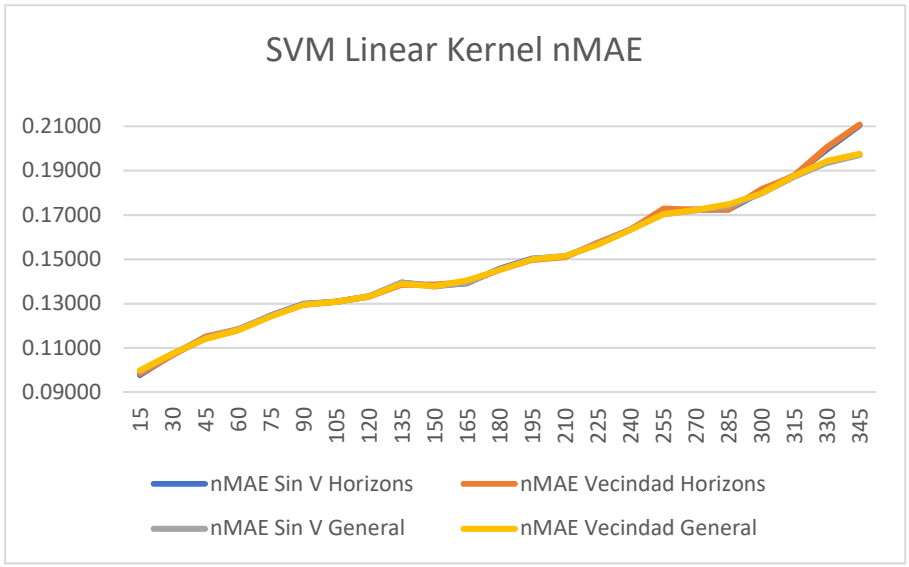

Ilustración 33: nMAE chart for SVM with linear kernel

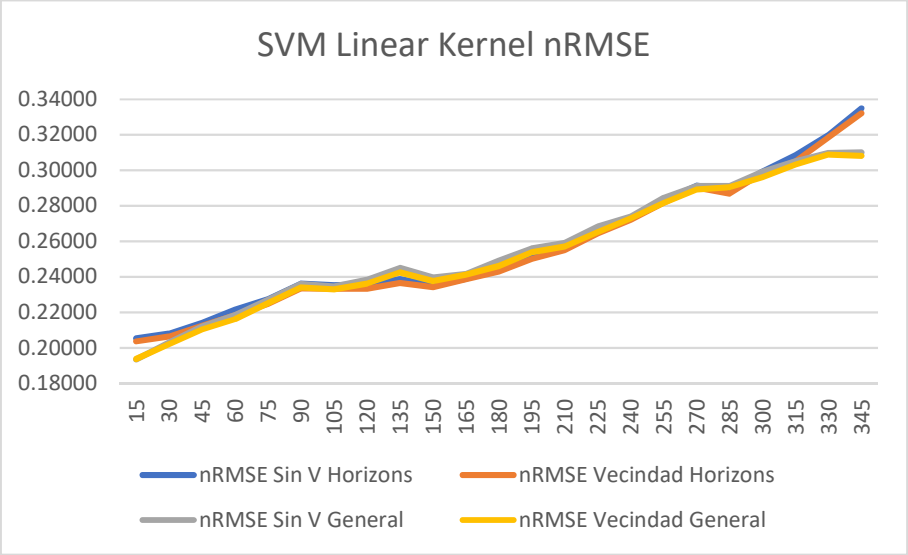

Ilustración 34: nRMSE chart for SVM with linear kernel

In the illustration 33, it could not be appreciated, but in the data the normal model shows a better result, being a thousandth better but if a study of the other metric is made, as seen in illustration 34, it can be observed that the neighbourhood model obtains better results. This may explain why, even though the mean of the error in the general model is smaller, there are errors with greater distance from the prediction, which is what penalizes the second metric, so in this case, the neighbourhood model continues to show better results. We can also see a trend like the previous example, although in this case the general approach shows better results than the horizons approach.

SVM with radial kernel

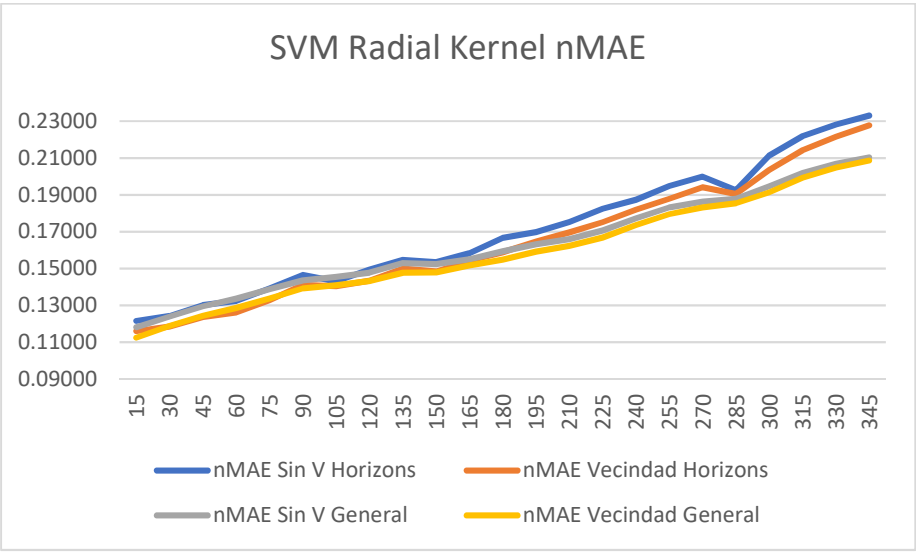

Ilustración 35: nMAE chart for SVM with radial kernel

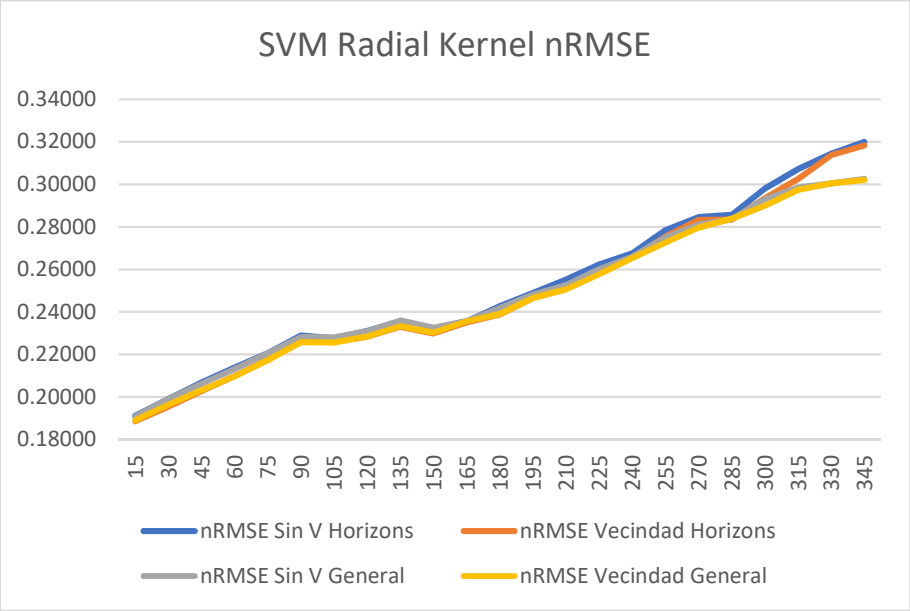

Ilustración 36:nRMSE chart for SVM with radial kernel

In this new example, as shown in figure 35, neighbourhood models show better results than general models. As in the previous example, the general approach shows better results in the first graph, but if the figure 36 is analysed, it can be seen that the horizons approach shows better results in the closest horizons and as it moves further in time, the General approximation shows better results.

Random Forest

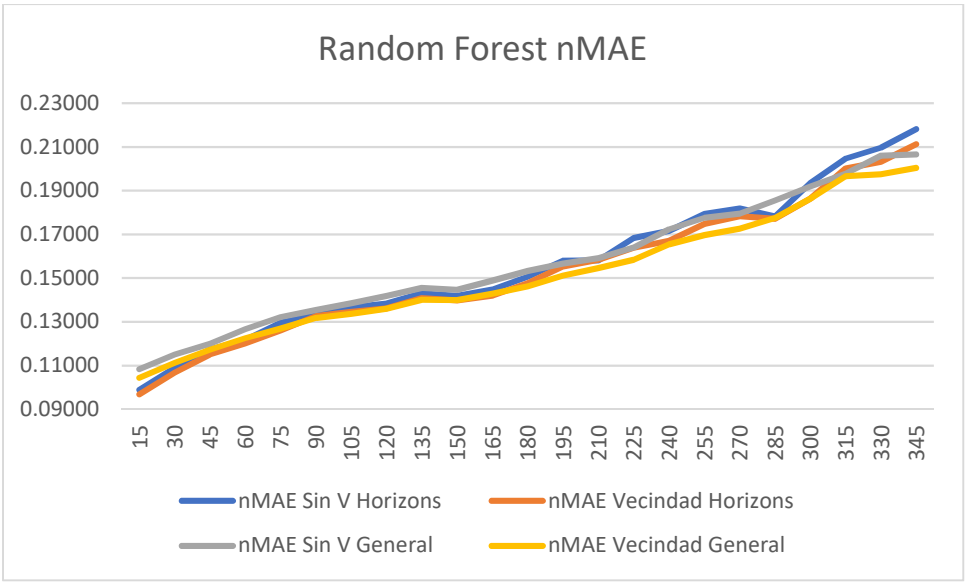

Ilustración 37: nMAE chart for Random Forest

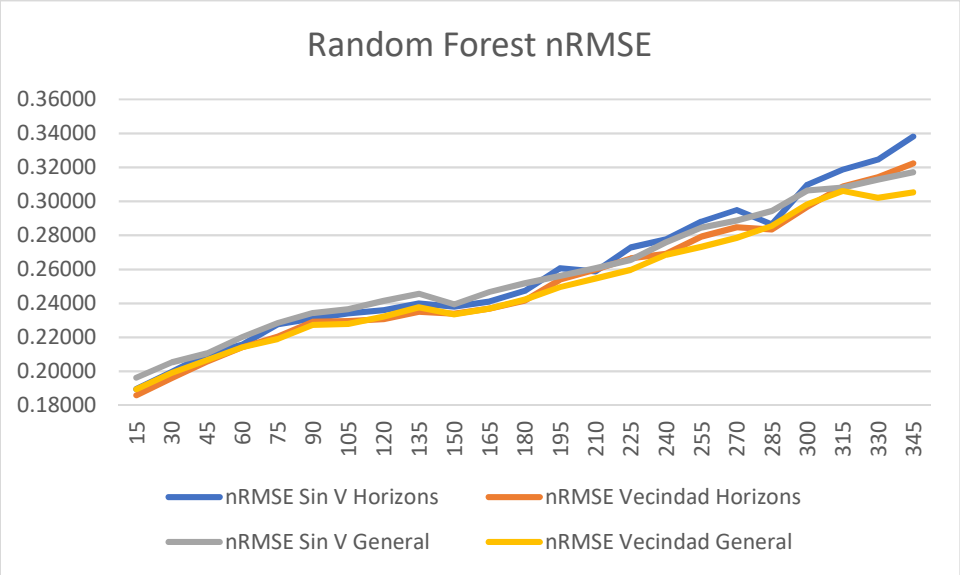

Ilustración 38: nRMSE chart for Random Forest

In this last example, as can be seen in illustration 37, the neighbourhood model obtains better results in both approaches, with a slightly better approach by horizons. It can be seen in illustration 38 as in this example the error increases as they go away in time but both approximations are very even, where normally the horizons approaching began to worsen, this time the difference is slightly less in distant horizons.

Conclusions

After showing the results and being analysed, it can be concluded that in a systematic way the neighbourhood model shows better results. This means that it improves the general model used up to now, which is a satisfactory result for the case study.

Regarding the 2 approaches used, one general and the other by horizons, it can be seen how the horizons approach is usually better when predicting values in medium and close horizons, while the general approach obtains better results in distant horizons.

That the approach by horizons has worse results in distant horizons has an explanation. The predictions of the physical models are made up to 6 hours, this means that for example, a normal winter day will get dark around 6 PM in the afternoon. This means that readings made after 12 PM will begin to gradually lose information by not having the final horizons. This implies that the final horizons tend to have less data, so the predictions will tend to be less accurate with this approach. The general approach can compensate this with all the data, but the model by horizons is unable to do it.

With this we can conclude that there is an improvement with respect to current approaches and that it is worth giving this approach a chance. Although the improvement has been marginal, this does not imply that the results are bad, but that we need to make a more in-depth study of the neighbourhood model.

Regarding the objectives set, all of them have been met, since otherwise it would not have been possible to complete this project. The sample of results is the culmination of the achievement of the marked objectives.

Anexo D- Bibliografía y Referencias

[1] «Informe del Sistema Eléctrico Español 2017 | Red Eléctrica de España». [Online]. Disponible en: http://www.ree.es/es/estadisticas-del-sistema-electrico-espanol/informe-anual/informe-del-sistema-electrico-espanol-2017

[2] «Las energías renovables en el sistema eléctrico español 2017 | Red Eléctrica de España».[Online]. Disponible en: http://www.ree.es/es/estadisticas-del-sistema-electrico-espanol/informe-de-energias-renovables/informe-2017

[3] «How do Photovoltaics Work? | Science Mission Directorate». Disponible en: https://science.nasa.gov/science-news/science-at-nasa/2002/solarcells

[4] J. Huertas-Tato, R. Aler, F. J. Rodríguez-Benítez, C. Arbizu-Barrena, D. Pozo-Vázquez, y I. M. Galván, «Predicting Global Irradiance Combining Forecasting Models Through Machine Learning», en *Hybrid Artificial Intelligent Systems*, 2018, pp. 622-633.

[5] L. Nonnenmacher y C. F. M. Coimbra, «Streamline-based method for intra-day solar forecasting through remote sensing», *Solar Energy*, vol. 108, pp. 447-459, oct. 2014.

[6] P. Jimenez *et al.*, «WRF-SOLAR: Description and clear-sky assessment of an augmented NWP model for solar power prediction», *Bulletin of the American Meteorological Society*, vol. 97, pp. 1249-1264, ago. 2016.

[7] C. Arbizu-Barrena, J. A. Ruiz-Arias, F. J. Rodríguez-Benítez, D. Pozo-Vázquez, y J. Tovar-Pescador, «Short-term solar radiation forecasting by advecting and diffusing MSG cloud index», *Solar Energy*, vol. 155, pp. 1092-1103, oct. 2017.

[8] C. Rigollier, M. Lefèvre, y L. Wald, «The method Heliosat-2 for deriving shortwave solar radiation from satellite images», *Solar Energy*, vol. 77, n.º 2, pp. 159-169, ene. 2004.

[9] C. Rigollier, O. Bauer, y L. Wald, «On the clear sky model of the ESRA — European Solar Radiation Atlas — with respect to the heliosat method», *Solar Energy*, vol. 68, n.º 1, pp. 33-48, ene. 2000.

[10] P. A. Devijver y J. Kittler, Pattern Recognition: A Statistical Approach. Englewood Cliffs, N.J: Prentice Hall, 1982.

[11] C. Cortes y V. Vapnik, «Support-vector networks», *Mach Learn*, vol. 20, n.º 3, pp. 273-297, sep. 1995.

[12] A. J. Smola y B. Schölkopf, «A tutorial on support vector regression», *Statistics and Computing*, vol. 14, n.º 3, pp. 199-222, ago. 2004.

[13] T. K. Ho, «Random decision forests», en *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278-282 vol.1.

[14] «The Incredible Growth of Python | Stack Overflow», Stack Overflow Blog, 06-sep-2017. [Online]. Disponible en: https://stackoverflow.blog/2017/09/06/incredible-growth-python/

[15] «About us — scikit-learn 0.19.2 documentation». [Online]. Disponible en: http://scikit-learn.org/stable/about.html#funding

[16] «Sobre las licencias - Creative Commons». [Online]. Disponible en: https://creativecommons.org/licenses/?lang=es_ES.

[17] «History and License — Python 3.7.0 documentation». [Online]. Disponible en: https://docs.python.org/3/license.html.

[18] F. Pedregosa et al., «Scikit-learn: Machine Learning in Python», Journal of Machine Learning Research, vol. 12, p. 2825−2830, oct. 2011.